推酷

# 编程狂人

programming madman

NO. 32

# 关于推酷

推酷是专注于IT圈的个性化阅读社区。我们利用智能算法, 从海量文章资讯中挖掘出高质量的内容,并通过分析用户 的阅读偏好,准实时推荐给你最感兴趣的内容。我们推荐 的内容包含科技、创业、设计、技术、营销等内容,满足 你日常的专业阅读需要。我们针对IT人还做了个活动频道, 它聚合了IT圈最新最全的线上线下活动,使IT人能更方便地 找到感兴趣的活动信息。

# 关于周刊

《编程狂人》是献给广大程序员们的技术周刊。我们利用技术挖掘出那些高质量的文章,并通过人工加以筛选出来。每期的周刊一般会在周二的某个时间点发布,敬请关注阅读。

本期为精简版 周刊完整版链接:http://www.tuicool.com/mags/53ba57b5d91b141eb6152331

欢迎下载推酷客户端体验更多阅读乐趣

# 目录

1.腾讯开源平台上线，聚合腾讯开源项目

2.2014年中国软件开发者调查报告

3.为什么每个前端开发者都要理解页面的渲染?

4.百度FEX刘平川：做最专业的前端

5.数据库的最简单实现

6.TokuMX使用小计

7.怎么评价淘宝 UED 的 Midway Framework 前后端分离?

8.游戏软件开发，硬件的架构影响有多大?

9.面试总结

10.专访QQ大数据团队，谈分布式计算系统开发

11.前端之Android入门(6):屏幕适配

# 腾讯开源平台上线，聚合腾讯开源项目

作者:王子殿下

腾讯公司上线“腾讯开源平台”，用来展示腾讯公司开源的优秀项目，主要包括追风移动加速、云排序Ctaxis、安卓性能测试工具APT、跨平台图形编译工具Koala、腾讯分布式数据仓库TDW、协程基础库Libco。

腾讯公司上线 “腾讯开源平台”，用来展示腾讯公司开源的优秀项目。


腾讯开源平台： http://code.tencent.com/。

目前该平台列出了6款腾讯公司的开源项目，分别有：

- QcloudMna：追风移动加速，腾讯云针对移动端应用（APP、游戏）推出的加速产品；
- Ctaxis：腾讯云计算平台提供的排序服务，云排序系统采用可扩展的分布式存储方案，具有支持海量数据、排序规则可灵活调整、数据上报和查询方便等特点；
- APT：安卓平台高效性能测试工具，适用于开发自测和定位性能瓶颈，以及性能基准测试、竞品测试；

- Koala：一款跨平台的预处理器语言图形编译工具，支持Less、Sass、CoffeeScript、Compass框架的即时编译；
- TDW：腾讯分布式数据仓库，基于Hive和Hadoop之上构建的数据仓库系统；
- Libco：协程基础库，目前已经覆盖整个微信后台。

由此我们可以看出，腾讯已经在开源上迈出了第一步，值得鼓励和肯定。当然，相比百度、阿里等公司在开源项目数量、与开源社区间的关系、内部开源文化培养等方面，腾讯在此方面还有待加强。

原文链接: http://code.csdn.net/news/2820473

# 2014年中国软件开发者调查报告

作者:魏兵

摘要：2014年5月-6月，CSDN携手《程序员》杂志发起的“中国软件开发者大调查”活动得到了广大开发者的大力支持。通过对调查数据分析形成调查报告将于近期上市，7月刊《程序员》杂志刊载了报告内容精华，本文为节选内容。

十年前CSDN携手《程序员》杂志首次发起的“中国软件开发者大调查”活动中，本刊记者用《不安分的2004和震荡的2005》作为调查报道的标题。而十年之后，在《2014年中国软件开发者调查报告》中，展现在我们面前的，对软件开发者来说，这是最好的时代。

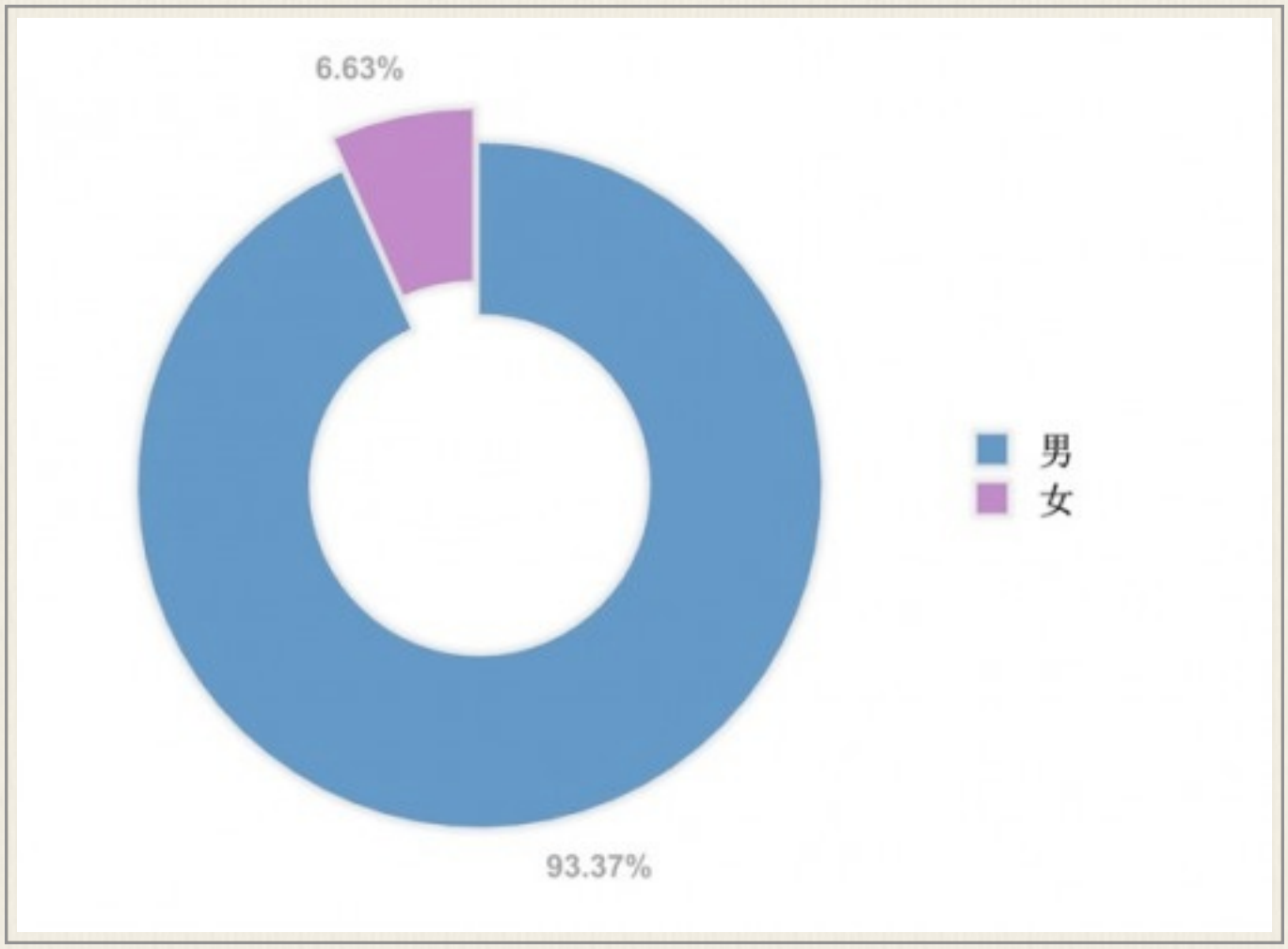
随 着互联网迅速发展，云计算、大数据由“热点”到落地，使得软件开发者不再为数据存储、数据分析而焦头烂额；互联网进入移动时代，Android、iOS平 台给予了软件开发者更大的舞台空间；硬件设备的发展，这十年间出现了诸如iPad、Kindle等手持阅读设备，使得学习渠道更加便捷；开源环境日趋成 熟，国内外开源共享平台发展迅速，开源软件及库造就了诸多传奇产品；在线教育迅速发展，在国内外各类公开课平台上，免费的计算机课程比比皆是……

## 开发者篇

十年光阴，岁月更替，说是变化巨大，但仍有些亘古不变的东西。比如，男性年轻人永远都是主力军。调查数据显示，男性软件开发从业者占总数的 93.37%，其中21~30岁占总人数的82.73%，而21~25岁的更占了总人数的51.2%。同时，数据显示，女性软件开发者占比并非一直很低， 在1~3年从业时长的开发者中，女性占比为6.8%，而随着从业时长增加，占比逐渐降低，达到15年以上时，几乎没有女性存在。

随着中国教育普及度增长及高校的不断扩招，软件开发者的学历水平在逐渐提升。本次调查参与者中，64.09%拥有本科学历，硕士及以上学历占比也超过15%。

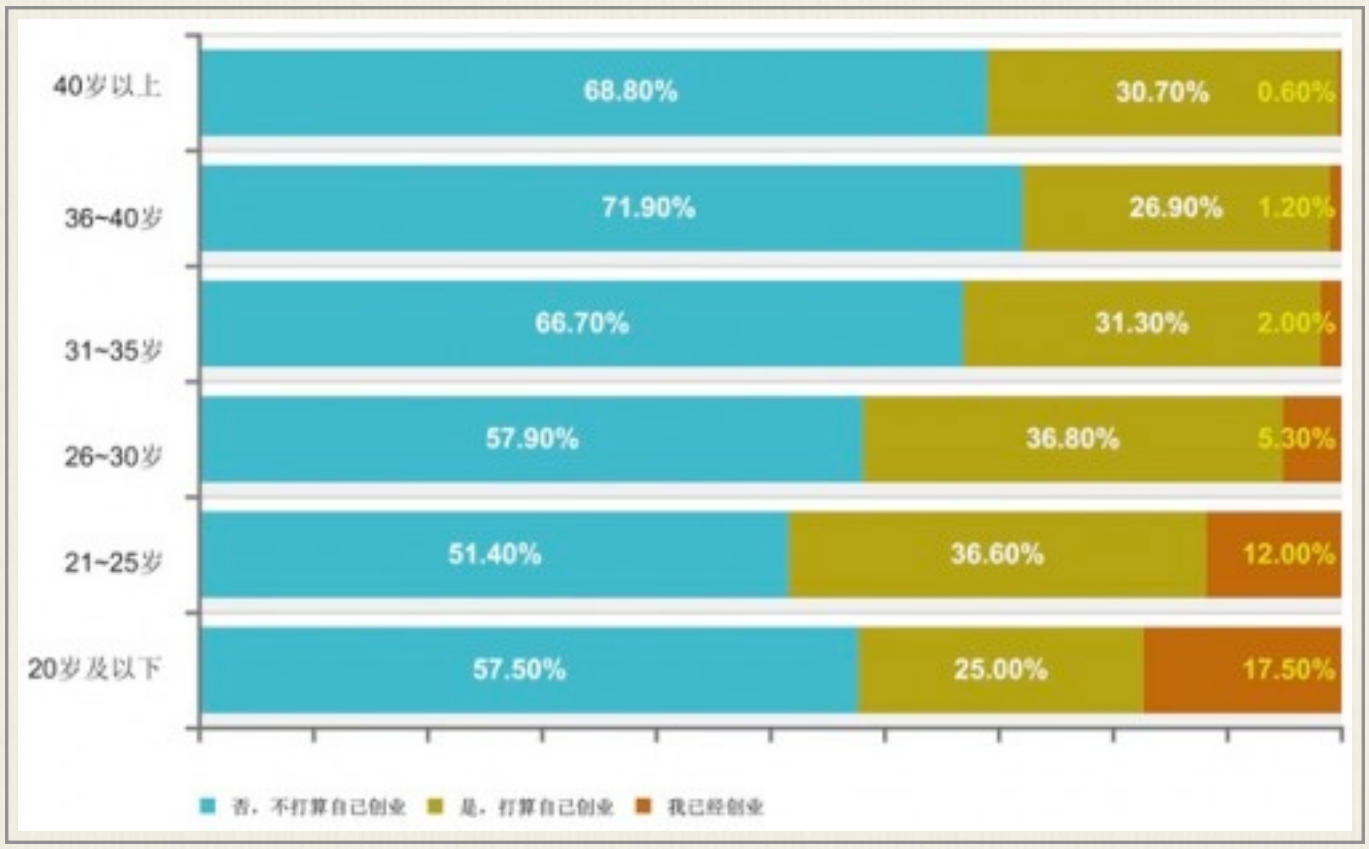
参 调者来自地区与国内IT发展环境态势相符合，北京、上海、深圳、广州与成都的参调者占比总和为47.74%，其中北京占比更是超过20%。一直以来，北上 广科技发展迅速，所以十年来它们一直是开发者最愿接受的工作城市。得益于智能硬件行业发展迅速，软硬件结合趋势明显，深圳在近几年也成为不少软件开发者乐 于选择的栖息地之一。




开发者男女比例分布

30岁，对于不少软件开发者来说，是一道坎儿。网上时常会出现“程序员30岁以后做什么”这样的讨论。与十年前相比，30岁以后 开发者继续写代码的意愿在降低。在CSDN此前的调查中，有62.11%的开发者表示在30岁以后不会再从事开发工作。而十年前的调查中，当时年龄在35 岁左右的软件人，基本都在软件公司“担当着成熟、理性、有主见的软件开发带头人角色”。

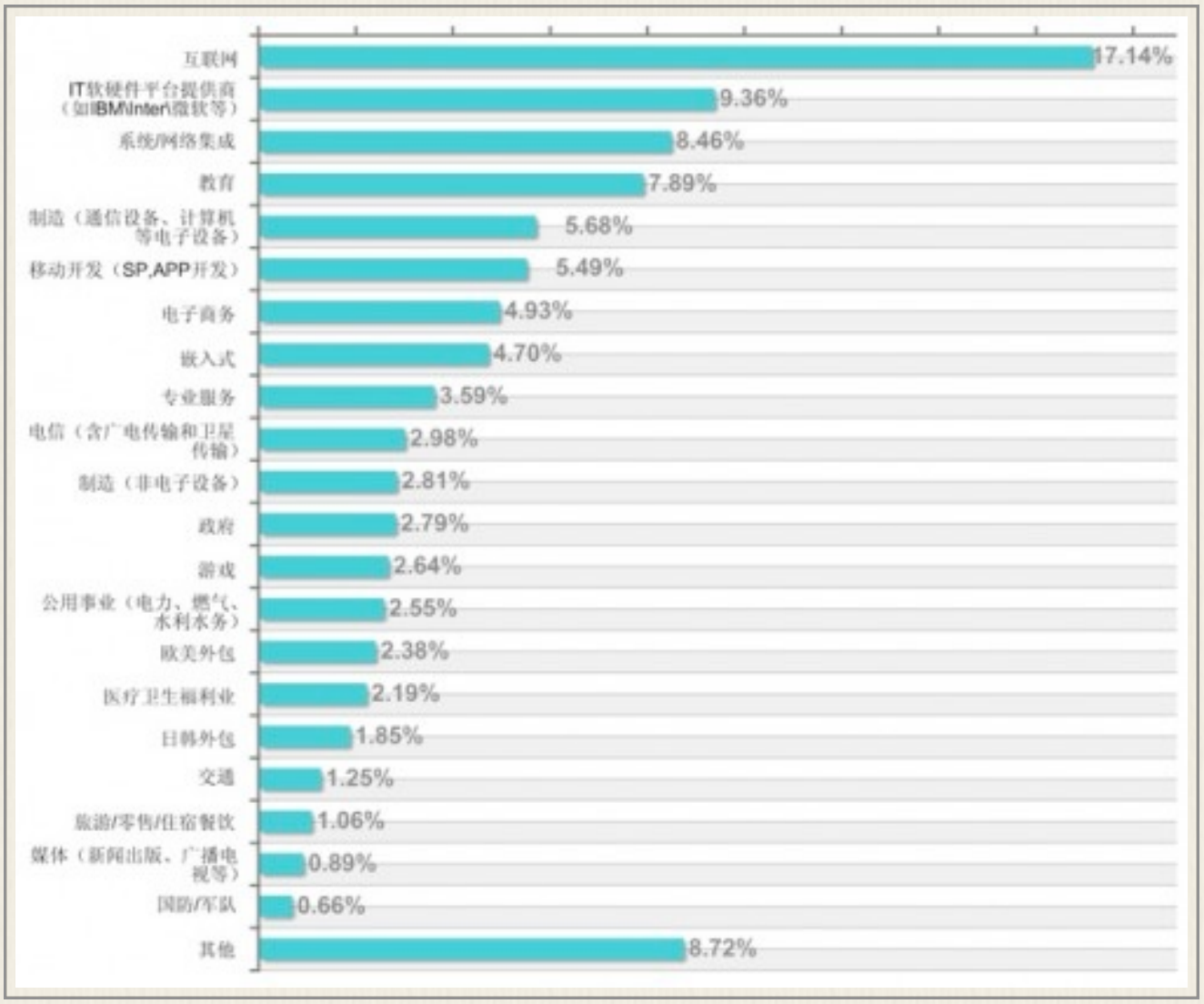
本次调查中，我们发现软 件开发者在30岁以后，计划创业或已经创业的比例超过了40%。而在30岁以下，这一数字不超过33.3%。同时，数据显示，从业时间越长，计划创业或已 经创业的比例越高，其中有10~15年工作经验的开发者创业意愿最强烈，占比超过50%。相对应，此阶段的开发者年龄都在35~36岁左右，已经积累了足 够的经验，拥有技术优势、懂产品、懂趋势，进行创业的话，拥有了其他人不具备的优势，但“一将功成万骨枯”，成功者的背后倒下过多少人我们也需要在心中掂 量掂量。




不同年龄段开发者创业态度

## 产品篇

即使科技发展迅速，软件 在企业中的应用越来越广泛，但调查数据显示，大多数软件开发者仍然来自IT企业，其中互联网、IT软硬件提供商（如IBM/Intel/微软等）、系统 /网络集成三个细分行业占比位于前三，分别为17.14%、9.36%及8.46%。所在公司研发人员规模也不尽相同，其中有24.9%参调者表示公司研 发人员规模不到9人，研发人员规模在10~99人之间的公司占35.33%，在100~499人之间的公司占19.03%，而研发人员规模在500人以上 的公司超过20%。




开发者公司所在行业分布

产品上，互联网后端（服务器端）产品是软件开发者主要开发的软件类型，占比为24.09%。绝大多数行业的软件产品都离不开服务器端的支撑，尤其在 互联网、电子商务及金融（银行/证券/保险）行业中，互联网后端（服务器端）软件开发占比较大，参调人员中，这三个行业中分别有46.4%、40.3%和 33.7%的软件开发者在从事互联网后端（服务器端）软件产品的开发。

除了互联网后端（服务器端）产品外，企业级应用（ERP/CRM/SCM/BPM等）和移动应用客户端也是开发者主要开发的产品类型，占比分别达到17.27%和14.01%。

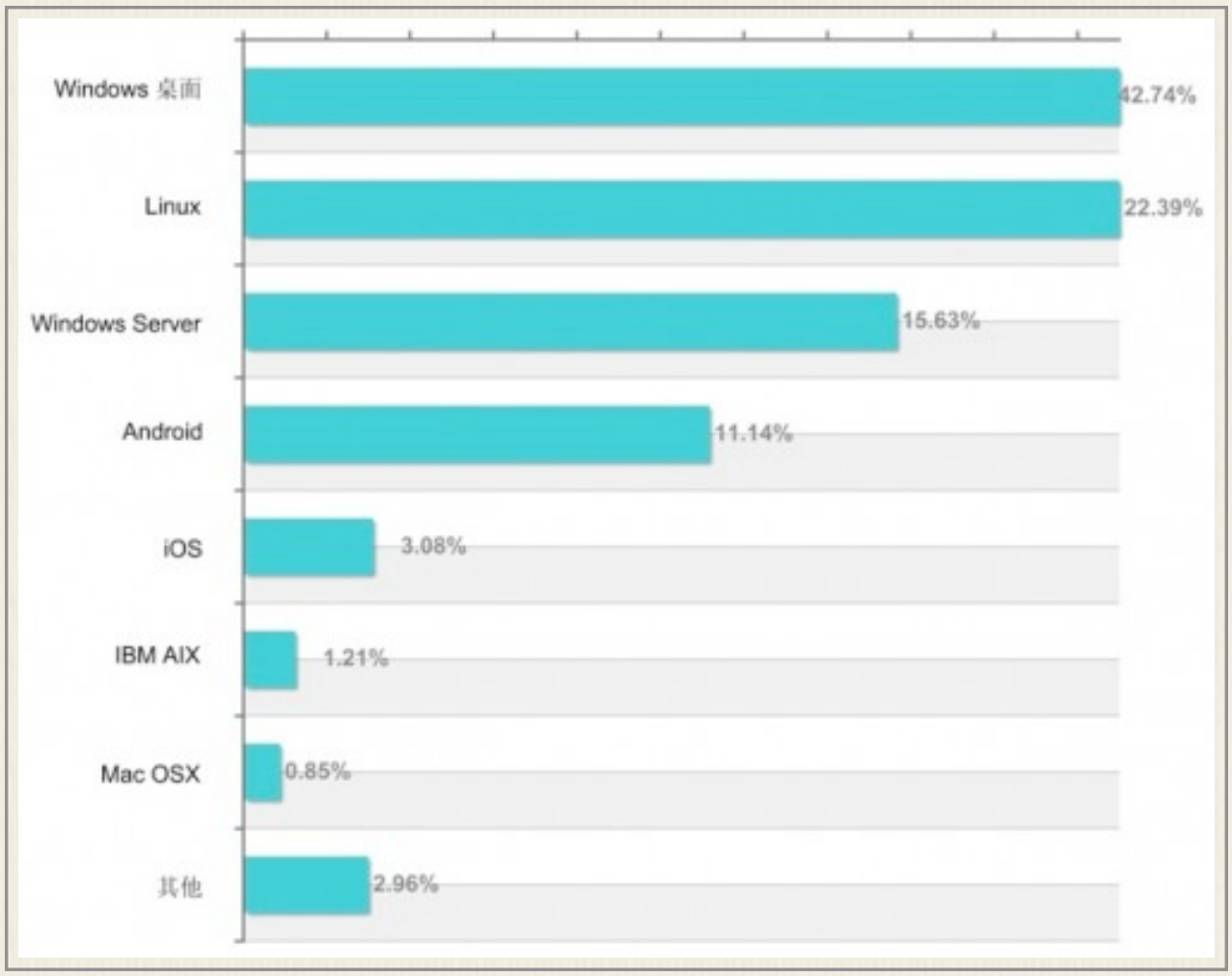
近 几年，移动互联网发展已经逐渐成熟，凭借移动应用开发一夜成名的软件开发者不时出现。调查数据显示，移动应用产品在各行业表现渐有超越互联网前端及桌面客 户端应用产品之势。尤其在游戏行业中，移动应用客户端产品开发人员占总人数的42.7%，这与国内移动游戏市场发展的火热息息相关。另外，得益于移动互联 网对基于地理位置信息服务（LBS）发展

的促进，旅游/零售/住宿餐饮行业中，移动类产品相比其他行业同样略高，达到17.3%。相信在未来几年，移动应 用类产品开发仍然会是热点，软件开发者需求仍然持续增长。

## 操作系统篇

根 据StatCounter提供的数据显示，2014年5月，Windows 7在中国市场的份额为49.92%。即便Windows XP在2014年4月8日正式退役，其市场份额仍然居高不下，为41.35%，保守估计，Windows XP在中国市场的用户不低于2亿。

与 此相应，调查中有42.74%软件开发者表示自己所开发的项目是面向Windows桌面操作系统，有15.63%软件开发者所开发项目主要面向 Windows Server系统。除此之外，面向Linux系统开发项目的开发者有22.39%。同时，移动互联网市场发展迅速，面向Android及iOS操作系统进行项目开发的开发者分别占11.14%与3.08%。参调开发者中，也有部分Mac OS X系统项目的开发者，占比为0.85%。




开发项目面向操作系统分布

5月16日，中央政府采购网发布了《中央国家机关政府采购中心重要通知》，其中第5点注意事项称，“所有计算机类产品不允许安装Windows 8操作系统”。相信这样一款禁令将会影响国内不少开发者的职业发展之路。“禁令”虽然只是针对国家政府机关，但同时也会影响其他政府机构或企业单位对操作 系统的选择，如国防/军队、交通及医疗卫生、金融（银行/证券/保险）、教育、制造（通信设备、计算机等电子设备）、电信（含广电传输和卫星传输）、公共 事业（电力、燃气、水利水务）、制造（非电子设备）及媒体（新闻出版、广播电视等）。而调查数据显示，有38.42%的软件开发者所在公司软件产品是面向 政府或以上可能受影响行业的。据此我们可以推测，在未来一两年，面向Windows系列操作系统开发产品的软件开发者群体比例将会降低，Linux系操作 系统软件产品或有机会出现一定增长。而随着移动智能设备的进一步普及，Android、iOS系统产品开发市场将持续增长。由于Android及iOS的 强势，其他移动操作系统在相当长的一段时间内将很难实现突破。

## 编程语言与平台篇

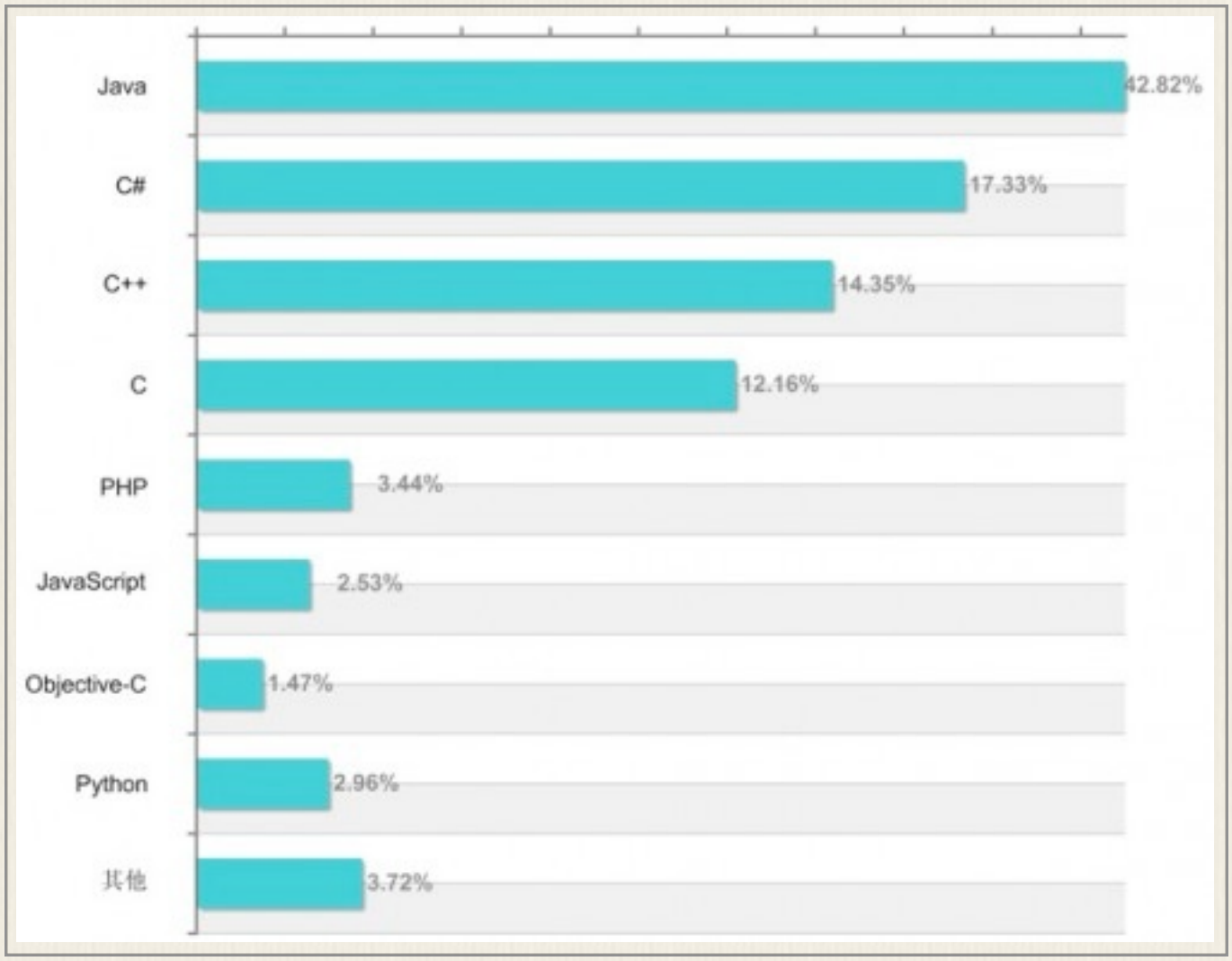
### 老而弥坚的Java

Java语言诞生已近20年，作为主流开发语言一直备受关注。本次调查中，42.82%的软件开发者表示Java是自己的第一编程语言，C#与C++紧随其后以17.33%、14.35%的占比位列第一编程语言的第二、三位。

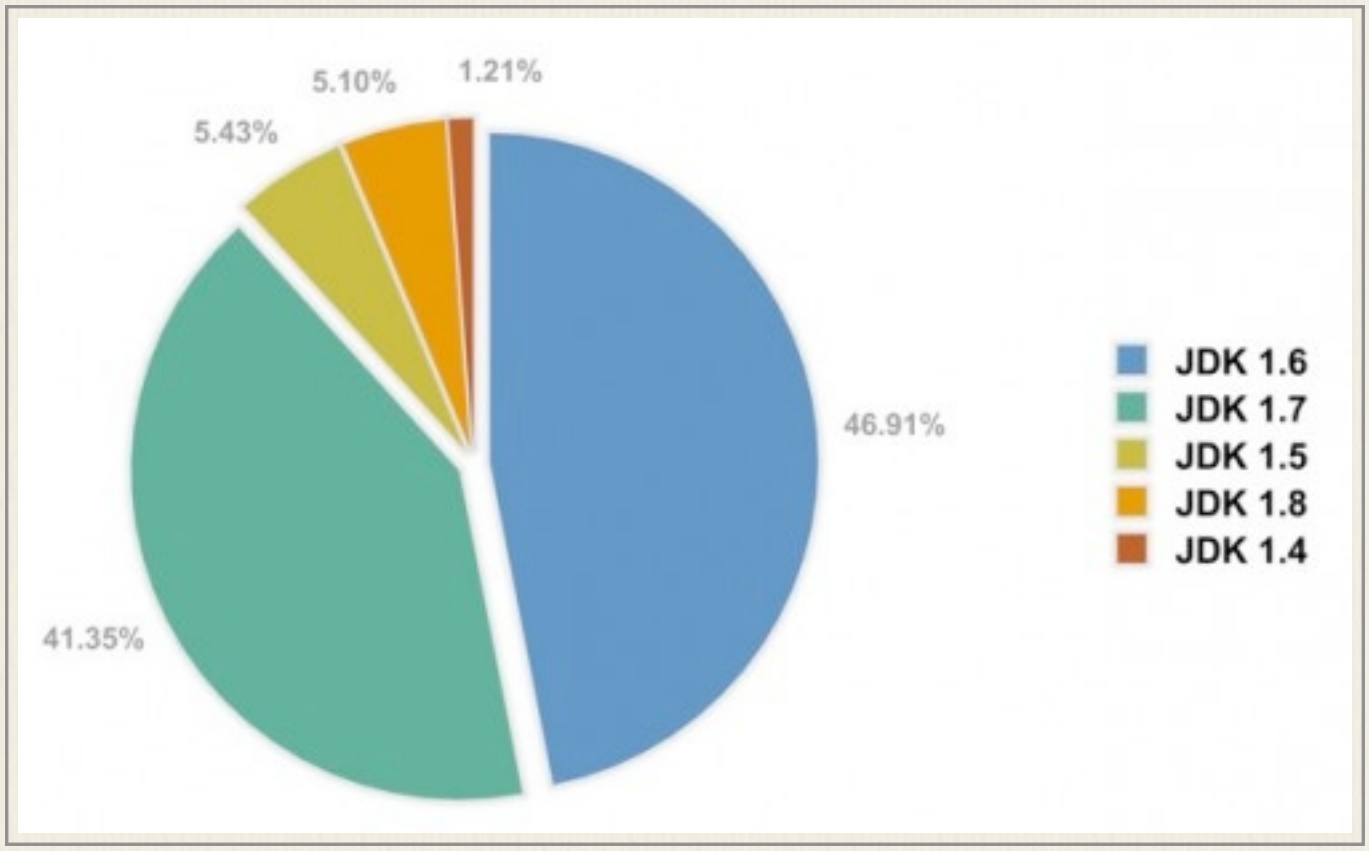
Oracle公司在3月18日发布了Java 8正式版，同期发布的还有JDK 1.8，新版JDK带来了不少新特性。但新技术在市场的普及仍需时日，调查结果显示，Java开发者中，目前使用最多的JDK版本是JDK1.6和 JDK1.7，两个版本使用率为88.26%。

Java开发主要使用的应用服务器是Tomcat，在开发者中受众比为75.71%，Oracle Weblogic有10.12%，而Jboss AS和IBM WebSphere分别只有4.35%和5.48%。在Web应用开发所采用的服务器端技术中，Java支持率最高。

Java开发人员使用的Java Web开发框架前两位是Spring MVC和Struts，占比分别为36.66%和25.79%，这两者的使用占总份额的62.45%。另外值得注意的是，Java开发人员中不使用框架的 比例达到了11.54%，自行开发框架的比例也达到了10.91%。




开发者第一编程语言分布情况




开发者目前采用JDK版本分布

Java 8的新特性在发布之后迅速成为了开发者非常关注的技术热点，除此之外，Struts+Spring+Hibernate、Hadoop及Spring MVC也是开发比较关注的技术热点。

没有太多的意外，绝大多数Java开发人员使用的开发工具是Eclipse，不过随着开发从业时长的增长，Eclipse份额有所减少，而Intelli JIDEA和Net-Beans比例有所上升。

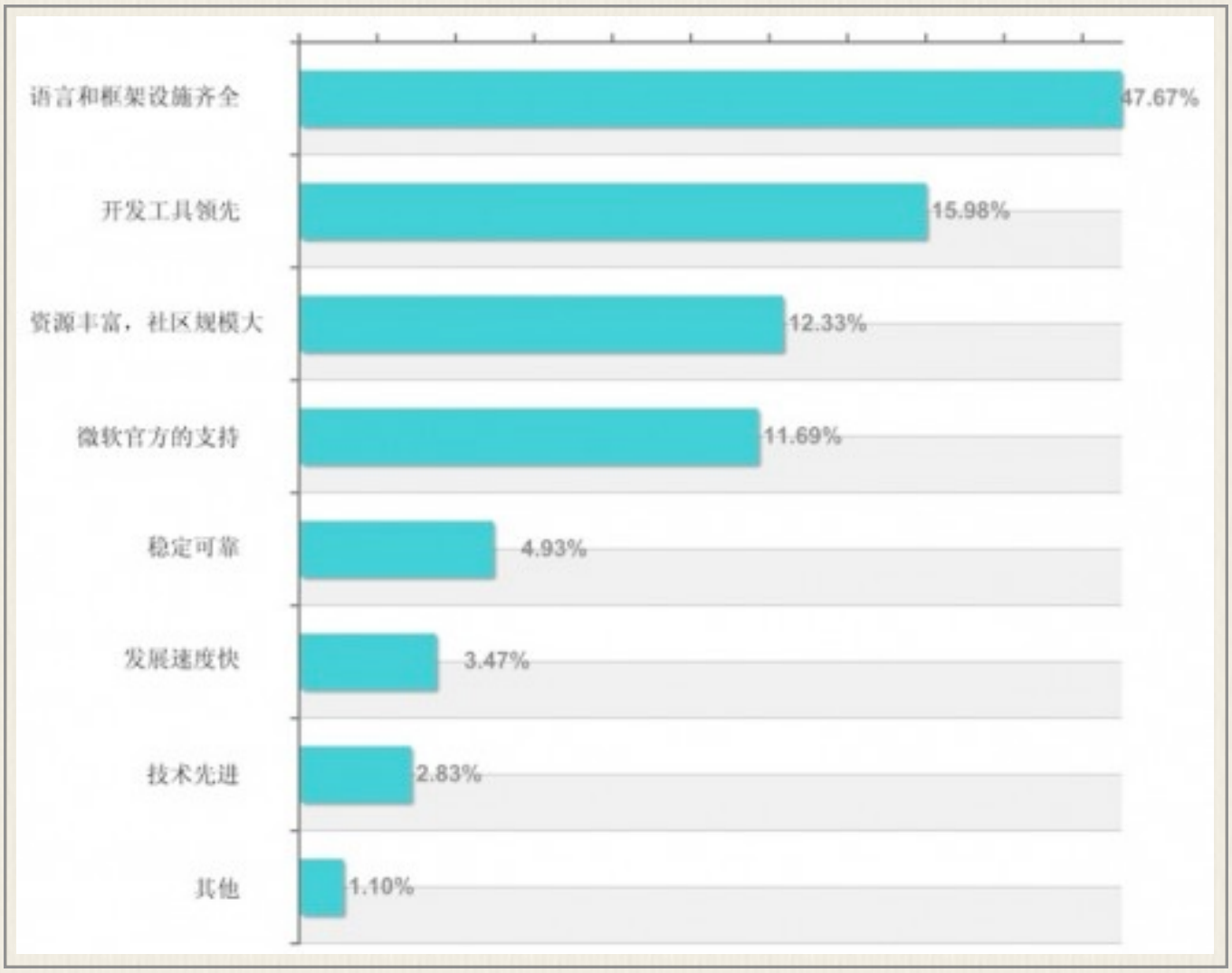
## .NET平台生态圈前景可期

4月，微软宣布成立.NET基金会，微软称基金会将为企业客户、开发者提供一个社区平台，进一步强化.NET生态圈。同时微软还将旗下24款.NET软件基于Apache 2.0许可协议开源。

本 次调查中，有23%的开发者表示从事.NET开发工作。其中82.56%开发者对.NET平台感到满意，语言和框架设施的齐全最令开发者满意，开发工具领 先、资源丰富以及社区规模大、微软官方的支持等同样是.NET令开发者满意的因素。当然，.NET也有令开发者不满意的地方，如限定在Windows平 台、性能资源耗用多等，其中限定在Windows平台的不满意率最高，达36.89%。

使用最多的.NET开发工具是Visual Studio 2013和Visual Studio 2008，这两者的比例占到了开发工具的81.83%。

.NET 开发人员最关注的技术分别是ASP.NET MVC、ASP.NET Web Pages、Windows Azure相关的云计算技术，其中ASP.NET MVC关注度为33.24%，ASP.NET Web Pages为14.61%，Windows Azure相关的云计算技术则为13.24%。而WPF技术、用Mono或Xamariam开发移动应用类也占据一些比例，分别是8.68%和 7.49%。




.NET技术最令开发者满意的地方

原文链接:http://www.csdn.net/article/2014-07-02/2820493

# 为什么每个前端开发者都要理解网页渲染?

译者:Moejser

今天我要将关注点放到网页渲染以及其重要性上。虽然已经有很多文章提到过这个主题了，但大部分信息都是零碎的片段。为了思考这件事情，我需要研究很 多信息的来源。这也就是为什么我觉得我应该写这篇文章的原因。我相信这篇文章对新手会很有用，并且对想刷新和巩固他们已经了解的东西的高手也同样适用。

渲染应该从最开始当页面布局被定义时就进行优化，样式和脚本在页面渲染中扮演着非常重要的角色。专业人员知道一些技巧以避免一些性能问题。

这篇文章不会深入研究浏览器的技术细节，而是提供一些通用的原则。不同浏览器引擎工作原理不同，这就使特定浏览器的学习更加复杂。

## 浏览器是怎样渲染一个页面的?

我们从浏览器渲染页面的大概过程开始说起:

1.由从服务器接收到的 HTML 形成 DOM（文档对象模型）。

2.样式被加载和解析，形成 CSSOM（CSS 对象模型）。

3.紧接着 DOM 和 CSSOM 创建了一个渲染树，这个渲染树是一些被渲染对象的集合（ Webkit 分别叫它们”renderer”和”render object”，而在Gecko 引擎中叫”frame”）。除了不可见的元素（比如 head 标签和一些有 display:none 属性的元素），渲染树映射了 DOM 的结构。在渲染树中，每一个文本字符串都被当做一个独立的 render-

er。每个渲染对象都包含了与之对应的计算过样式的DOM 对象（或者一个文本块）。换句话说，渲染树描述了 DOM 的直观的表现形式。

4.对每个渲染元素来说，它的坐标是经过计算的，这被叫做“布局(layout)”。浏览器使用一种只需要一次处理的“流方法”来布局所有元素（tables需要多次处理）。

5.最后，将布局显示在浏览器窗口中，这个过程叫做“绘制(painting)”。

## 重绘

当在页面上修改了一些不需要改变定位的样式的时候（比如background-color,border-color,visibility)，浏览器只会将新的样式重新绘制给元素（这就叫一次“重绘”或者“重新定义样式”）。

## 重排

当页面上的改变影响了文档内容、结构或者元素定位时，就会发生重排（或称“重新布局”）。重排通常由以下改变触发：

- DOM 操作（如元素增、删、改或者改变元素顺序）。
- 内容的改变，包括 Form 表单中文字的变化。
- 计算或改变 CSS 属性。
- 增加或删除一个样式表。
- 改变”class”属性。
- 浏览器窗口的操作（改变大小、滚动窗口）。

• 激活伪类（如:hover状态）。

## 浏览器如何优化渲染?

浏览器尽最大努力限制重排的过程仅覆盖已更改的元素的区域。举个例子，一个 position 为 absolue 或 fixed 的元素的大小变化只影响它自身和子孙元素，而对一个 position 为 static 的元素做同样的操作就会引起所有它后面元素的重排。

另一个优化就是当运行一段Jjavascript 代码的时候，浏览器会将一些修改缓存起来，然后当代码执行的时候，一次性的将这些修改执行。举例来说，这段代码会触发一次重绘和一次重排：

```
var $body = $('body');
$body.css('padding', '1px'); // 重排, 重绘
$body.css('color', 'red'); // 重绘
$body.css('margin', '2px'); // 重排, 重绘
// 实际上只有一次重排和重绘被执行。
```

如上面所说，访问一个元素的属性会进行一次强制重排。如果我们给上面的代码加上一行读取元素属性的代码，这个情况就会出现：

```
var $body = $('body');
$body.css('padding', '1px');
$body.css('padding'); // 这里读取了一次元素的属性，一次强制重排就会发生。
$body.css('color', 'red');
$body.css('margin', '2px');
```

上面这段代码的结果就是，进行了两次重排。因此，为了提高性能，你应该讲读取元素属性的代码组织在一起（细节的例子可以看JSBin上的代码）。

有一种情况是必须触发一次强制重排的。例如：给元素改变同一个属性两次（比如margin-left），一开始设置100px，没有动画，然后通过动画的形式将值改为50px。具体可以看例子，当然，我在这里会讲更多的细节。

我们从一个有transition的CSS class开始：

```
.has-transition {
   -webkit-transition: margin-left 1s ease-out;
      -moz-transition: margin-left 1s ease-out;
        -o-transition: margin-left 1s ease-out;
           transition: margin-left 1s ease-out;
}
```

然后进行实现：

```
//我们的元素默认有"has-transition"属性
var $targetElem = $('#targetElemId');

//删除包含transition的class
$targetElem.removeClass('has-transition');

// 当包含transition的class已经没了的时候，改变元素属性
$targetElem.css('margin-left', 100);
```

*// 再将包含transition的class添加回来*

*$targetElem.addClass('has-transition');*

*// 改变元素属性*

*$targetElem.css('margin-left', 50);*

上面的实现没有按照期望的运行。所有的修改都被浏览器缓存了，只在上面这段代码的最后才会执行。我们需要的是一次强制重排，我们可以通过进行以下修改来实现：

*//删除包含transition的class*

*$(this).removeClass('has-transition');*

*// 改变元素属性*

*$(this).css('margin-left', 100);*

*//触发一次强制重排，从而使变化了的class或属性能够立即执行。*

*$(this)[0].offsetHeight; // off-*
*setHeight*仅仅是个例子，其他的属性也可以奏效。

*// 再将包含transition的class添加回来*

*$(this).addClass('has-transition');*

*// 改变元素属性*

*$(this).css('margin-left', 50);*

现在这段代码如我们所期望的运行了。

## 实际的优化建议

汇总了一些有用的信息，我建议以下几点：

- 创建合法的 HTML 和 CSS ，别忘了制定文件编码，Style 应该写在 head 标签中，script 标签应该加载 body 标签结束的位置。
- 试着简化和优化 CSS 选择器（这个优化点被大多数使用 CSS 预处理器的开发者忽略了）。将嵌套层数控制在最小。以下是 CSS 选择器的性能排行（从最快的开始）：

1.ID选择器：#id

2.class选择器： .class

3.标签: div

4.相邻的兄弟元素： a + i

5.父元素选择器： ul > li

6.通配符选择器： *

7.伪类和伪元素： a:hover ， 你应该记住浏览器处理选择器是从右向左的，这也就是为什么最右面的选择器会更快——#id或.class。

*div * {...} // bad*

*.list li {...} // bad*

*.list-item {...} // good*

*#list .list-item {...} // good*

1.在你的脚本中，尽可能的减少 DOM 的操作。把所有东西都缓存起来，包括属性和对象（如果它可被重复使用）。进行复杂的操作的时候，最好操作一个“离线”的元素（“离线”元素的意思是与 DOM 对象分开、仅存在内存中的元素），然后将这个元素插入到 DOM 中。

2.如果你使用 jQuery，遵循jQuery 选择器最佳实践

3.要改变元素的样式，修改“class”属性是最高效的方式之一。你要改变 DOM 树的层次越深，这一条就越高效（这也有助于将表现和逻辑分开）。

4.尽可能的只对 position 为 absolute 或 fix 的元素做动画。

5.当滚动时禁用一些复杂的 :hover 动画是一个很好的主意（例如，给 body 标签加一个 no-hover 的 class）关于这个主题的文章。

想了解更多的细节，可以看一下这些文章：

1.How browsers work

http://taligarsiel.com/Projects/howbrowserswork1.htm

2.Rendering: repaint, reflow/relayout, restyle

http://www.phpied.com/rendering-repaint-reflowrelayout-restyle/

希望这篇文章能够对你有所帮助！

原译文链接: http://blog.jobbole.com/72692/

原文链接: http://frontendbabel.info/articles/webpage-rendering-101/

# 百度FEX刘平川：做最专业的前端

作者:图灵访谈

刘平川，百度前端基础技术团队FEX负 责人。从“有啊”和“乐活”到如今的FEX，一种创业的热情一直跟随着他。FEX的关键词包括开源，前端，全栈，和专业。虽然他们是基础技术团队，但是就 像“内部创业一样”，他们也需要时刻面对来自产品线的各种反馈。刘平川希望可以让FEX的技术影响扩展到百度，乃至整个行业。但他也坦诚说，“我们需要很 强的抗压能力”。

## 激情和责任

**你从什么时候开始编程的?**

**刘平川**：真正开始编程是上大学之后和朋友去做了一个网络监控系统，接触前端是当时工作关系写了几个脚本的UI开 源组件。后来2008年当时百度在做“有啊”的C2C，我对这个事情挺感兴趣。当时我虽然学了计算机，但实际上并没有真正从事这个行业，有些遗憾，于是我 就来到了北京，真正开始了前端工作。

**听说在此之后您还有一些创业的经历，能简单介绍一下吗?**

**刘平川**：我在百度两年半后从“有啊”拆分出去创业了，叫“乐活”。当时有很多“有啊”的同事也纷纷出去创业。大 家做C2C的时候聚到一起是因为有一个理想：把“有啊”这件事做成。虽然“有啊”最后没有做成，但是大家心里面都萌生了一个想法：下次一定要做出点事来， 这也是我愿意拆分出去创业的原因之一。

我在乐活创业一年多遇到很多问题，最后创业算是失败了，但是学了很多东西。最大的感触是创业不是有想法就能做得成，大多数的创业其实是水到渠成的事，因为需要天时地利人和。

后来我回到百度，加入了FEX团队。回去之前我就想，这个机会如果再晚一年可能我就不会考虑了，如果过了当时的年纪要真正再去花所有的时间和精力去做好技术是很难的。所以既然来了，我就希望能在FEX团队安静地把技术做好。

**刚来到百度FEX时情况是什么样的?**

**刘平川**：原来这个部门人还挺多的，整个百度所有的前端都在这个部门。在我回百度之前，前端已经拆分到每个垂直部 门了，还剩下大约一个60人左右的团队，支持公司很多产品线上前端基础技术上的业务。之前对这个部门了解程度有限，只知道这个部门有些不一样，也有很多技 术牛人。但是怎么样去做好，其实说老实话我不知道。我当时的想法就是希望能让这个团队在这个行业里面留下一些痕迹而已。

60个人已经是 3 个贴吧的规模了，如果有60个人，就不得不干60个人的活。这个规模至少得把好几个大产品线给接下来，我们暂时搞不来。人宁可少也不要多，人一多很多事情 没法做。我之后做了很大的调整，把一些方向，比如Flash拆分到垂直产品线去了，现在只有30人。这么做的目的只有一个，就是为了个人和部门的长期利 益。

我自己是很有激情去做好这个团队的，但很多人可能觉得我是一时兴起，也就干3个月热度吧。我希望给每个人一个舞台，团队平台做到足够大的影响力。一年过后，我可以告诉大家，今年团队的状态很好，已经不需要太多管理了。

**你在这里面的角色是什么?**

**刘平川**：我其实现在走的是M（管理），不是T（技术）。我的角色就是在各个负责人之间找我应该做的事情，为我们中长期的目标打算。比如我们团队的技术影响力，部门技术更长远的规划，我们明年做什么，怎么把部门做得更好。

去年底我们规范了运行机制，整体所有方向上包括技术规划和Review机制，还有各方向目标都公布出来。周报上面会写负责人是谁，衡量标准是什么， 需要多少人力，什么Level，在什么阶段交付，这个事情对产品线的价值是什么，都会写在上面。今年很多事情都比较上正轨，后面才考虑做技术品牌和技术规 划。这件事情团队其他同学，比如多益和牛尧付出很多，我只是做一些顺水推舟的事。

今年我们的重点是在线化，包括很多内容：资源打包优化、全流程优化、数据监控，等等。也包括我们把PC端的一些复杂应用Web化，这也是今年的一个 目标。在技术影响力上面，从我们的一些开源项目也好，跟业界的接触也好，我希望通过外部的资源和信息引入，推动内部更快地发展。

**你们和产品线的关系如此紧密，你的压力是不是很大?**

**刘平川**：我们各技术方向负责人压力确实会比较大，我们有机会可以接触到产品线里所有的人，产品线自然可以看到我 们的每一个阶段。每季度或每半年，他们要看到我们的产出是什么，如果没有很合适的交代的话，会受到产品线的质疑。我们也曾经遭遇过做了几个技术项目都没有 得到认可

的情况。可以想象这就像内部创业一样，创业需要很多的尝试。我们做得好了，每个产品线都会多给点资源。但如果我们做不好，就什么都没有。

无论是否做成我一直坚信的是，只有没做好的事，没有做不好的人，态度是首要的。一个团队可以做很多事，一个人可能这个事情做不好，做另外一个事情就好了。一个人发光发热需要时间，也需要一个巧合。

## 技术能解决什么问题

**你认为技术/工程师和产品的关系是什么样的?**

**刘平川**：我曾和一个朋友讨论技术，他的一套理论让我哑口无言。他说，设计模式就是人想出来的，也是一种想法的抽 象，你觉得设计模式做的就对了吗，我的代码乱是错的吗，如果我把技术问题解决了，你为什么要管我是怎么解决的呢，是否是你们对技术的评判标准有问题呢？后 来我想，是不是我们对代码优雅这个问题已经形成了一些思维定式，对于代码杂乱和提供问题快速解决方法的人有种不屑一顾呢?

我的一个朋友创业，他团队中的某个工程师，一个人仅用了一个月的时间搭完产品上线了。虽然技术的可维护性几乎没有，那你说这个人的技术是好还是不 好？这件事情没有办法用传统的思维来评判。原来的标准其实都是一种大公司式的评判标准。虽然他的代码别人无法维护，但是他做的事是有价值的，创造的经济价 值比传统科班出身的人多得多。后来其他工程师维护起来说代码太乱，用了3、4个月把代码改好。而这种能闯的人对创业公司其实更具有吸引力。

设计哲学这件事恐怕没有人能讲清楚。所以有些事不要太较真，要看这件事是为了什么。如果过两年这件事仍然对我们很重要，那就有价值，但是从大多数情况来看，一般过半年就不再重要了。

技术永远是一种手段，而不是目的。产品最终结果不在乎你技术有多牛，只在乎你能把产品带到什么高度，在这过程当中能解决多少技术问题。

**为什么要工程师自己担任产品经理?**

**刘平川**：我们没有产品经理，技术需求来源最主要从产品线来，我们需要了解工程师开发中的痛点，产品中的体验问题，找到合适的切入点开发出靠谱的技术方案。

现在每个技术方案负责人都是方案的产品经理。我要求团队的同学需要用这些做产品的思路来做技术，这样思考才会技术项目做得全面，系统，也更具有说服力。

一些比较具有前瞻性的项目我们可以先做原型，原型做完后再给到产品线中去，如果不回到产品线里去，我们肯定会做偏，技术会没有宿主。每个人都会有很多技术想法，如果你最后没有落实到产品线的话，就没有意义，所以我们做技术需要用产品的视角来做。

技术方案的推广上原来我们大多是强推着产品线工程师在用，甚至变成他们的KPI，这是有问题的。强推还不成熟的技术反而会引起负面效果。后来我们分 阶段对待，不用行政手段。把技术方案的使用广度作为一个阶段性的衡量标准，比如当某项技术方案质量做得足够好的时候，再去看一个周期内落地了多少产品线， 这样的评价才更会有意义。负责人需要每个阶段都有相对明确的定义，让每个工程师都能理解这个目标。

**技术人员身兼经理的必要性在哪?**

**刘平川**：基础技术的团队与产品业务团队有很大的不同。以前我带过几个业务团队，我个人的感觉是产品业务团队的运行是由产品业务来驱动的，经理协调“人”的时间比较多。

而在基础技术团队，是技术驱动项目，用产品思路做事，没有职能体系之间的协调会影响效率的问题。我们很少开会，每周固定的会议只有部门例会。团队之间技术讨论是最多的，所以我们会议基本都是在技术讨论与评审，我也需要参加，还要拍板决策。

如果基础技术团队的经理只负责协调人而不管技术还会有一个问题。由于经理没有办法判断技术的趋势，缺乏技术创新，团队的技术视野会逐渐受限。这样的 经理思维方式也不会跟一线工程师在一个频率上面，沟通不了。所以我认为基础技术团队的经理应该和高工有同样的职能，经理就是拥有行政权利的工程师。

**您曾经提过技术的止盈止损这个概念，是指什么?**

**刘平川**：技术既需要止损也需要止盈。很多技术项目可能当你做到一定程度之后，会发现它的意义已经不是那么大了，这个时候就应该果断地止盈或止损。

比如说编辑器，我们的UEditor编辑器在行业里面，至少在国内还算可以了，现在很多外部公司也都在用，我们每天都会收到问题反馈邮件。但我从去 年就在考虑，我们这个组的方向不能只限于做编辑器，应该把这个组的视野拓宽一点。是不是可以将现有PC本地端有的东西，放在Web上面去试试，我们认为 Web有这种能力，可以解决很多现有阶段Web的复杂功能。在今年，原来的富文本方向也转变成富应用方向，目前百度脑图、矢量可视化公式编辑也都是这个方 向的产出。

止盈止损是经济学里的一个概念，即是适可而止。放在技术上大概的意思就是像编辑器那样的成熟项目，可以暂时减少人力放缓发展。把人力资源抽调配到目 前更具有创新性的其他技术或虽不成熟但需要快速突破的方向上。我们可以一直抱着旧的东西不放，这块技术可以让我们活下去，但我一直认为，如果只做那一块技 术而不愿意去扩展，其实就太小看我们自己了。

**今年您参加了韩国举办的WWW年会，有什么见闻?**

**刘平川**：今年WWW大会在韩国首尔，主要的赞助公司是Naver，他们不仅有搜索引擎，还有一个叫Line的App，就是微信的韩国版。在韩国Naver就是百度+腾迅的产品架构，国内很多大互联网公司都“抄”过这个公司的产品。

我们国家的互联网公司在国外的知名度还是不高，更谈不到影响。与会场的一些参会人聊天时发现，他们不太清楚中国的互联网情况，也不知道百度、淘宝。后来我们提到百度是本次赞助商之一，他们才知道百度。 在韩国找路线的时候没法用百度地图，只能用Google。用它可以通过中文搜索酒店景点名字，包括公交路线也都能找到。如果没有Google 地图的话，我们几个没有一个人能在首尔出行，这给我触动很大。真正做得好的技术可以利用到很多行业，影响到很多人。

## FEX

**您能简要介绍一下FEX正在从事的开源项目吗?**

刘平川：开源在FEX是一种心态，接受review与check，接受反馈，希望用UGC思路把技术项目做得更好。目前我们也是这么践行的，从项目代码到面试题，以及博客和style guide，都接受Pull Request和Issue。

现在我们的开源技术项目大家熟知的有七八个，FIS、编辑器、脑图、公式、HTML5上传组件、移动组件。暂时都是工具和组件类偏多，目前star总量有近2000，fork也有大几百。

KityMinder是我们最新推出的项目，它是一个在线脑图编辑器，已经和百度云盘打通了。从技术角度来看，是因为之前我们有一定的矢量技术基础，所以初期脑图原型我们开发时间并不多，1个人不到1个月的时间开发完成上线，现在也在接受反馈，不断迭代。

上线以后，从Web内容消费与创作的产品思路来看 ，云盘解决了文件存储的问题。而文件内容，例如办公系列，教育图形公式的一些内容创作，还有很大的可在线化空间。我们做的话，既能沉淀技术，还可以帮百度 做很好的内容创作工具，落地到阅读、贴吧、知道、百科或云，甚至可以诞生出一个完整的创作平台。从对脑图的反馈来看，大家非常希望出现更多Web形式的全 端在线查看、编辑与分享的工具。

**FEX上最受欢迎的项目FIS，现在的发展状况怎么样？都有哪些公司在用？都有什么人在贡献代码?**

刘平川： FIS 的使用人群可大致分成三类。

第一类是百度内的工程师，随着产品线使用广度的不断扩宽，FIS已成为公司大多产品线前端开发的标准。我们把技 术服务做好，有问题第一时间解决；一旦有新产品线使用，我们会有同学跟进到产品线里进行培训。公司内反馈最多，我们也最为关注，我们会将他们的反馈抽象放 到FIS上，慢慢积累，使得FIS从量变到质变。公司内其他团队为FIS贡献不少，比如即将上线的NodeJS框架，是与文库同学一同合作开发的，可以说 FIS是在所有产品线的工程师帮助下才能走到今天。

第二类是从百度离开后将FIS转而带到其他公司的工程师，这是对FIS的认可。他们认为它可以为新的环境创造价值，解决新环境的问题。现在至少6家公司都是这样，同时他们也会对FIS贡献部分代码。

最后是通过国内外的技术杂志、讲座上看到对FIS技术的介绍，感兴趣并使用的工程师。这些工程师从了解到使用的周期相对会比较长，目前贡献的代码不多。但我们现在也很重视这样的人群，所以对文档不断维护，并持续更新，而官网最近也把FIS重新定位成前端工具和用工具及框架生成的解决方案，同时增加了使用解说视频，帮助大家更快地了解和使用FIS。

**FEX名字的由来?**

**刘平川**： FE代表前端（front-end），X代表了每个人都能独当一面，不仅所有事都了解一些，而且还有一个专长。就像X战警一样，每个人都有自己独特的能力，但是作为团队可以一起把事情做得更好。

**在网站上可以看到FEX的口号是“Best or nothing”，这看起来是一个比较有野心，也比较有风险的情况。你们怎么平衡跟现实之间的关系?**

**刘平川**：最早我们自己也没想清楚怎么样定位自己。无论是基础技术组还是通用组，感觉都不能很好地体现我们的能力。去年经过半年时间，我们逐步精简后把“专业”作为我们的定位，所有的技术项目、文章与践行都围绕“专业”来做。

与产品线合作，我对团队同学有几个要求：第一，我们一定要在这个方面做得比他们更专业更全面；第二，别人能做到的我们也要能做到；第三，我们也要去做业务，不要说只做基础技术部分，业务基础一样要做专业。

**你们团队里面的内部构成是什么样?**

**刘平川**：目前一共是4个大的方向：

- **工程架构和流程优化**，能力上可以理解为“全栈”。包括百度几乎所有产品线在用的FIS、静态资源自动优化打包优化、持续集成、全流程优化等，同时也在一定程度上承载我们其他方向组件与工具的输出。
- **监控评估**，包括现在几乎所有的前端数据的收集与监控，比如前端性能的监控，浏览器新特性的检测，异常的监控，XSS检测。这其中还包括

评估，也就是针对监控和产品核心指标的关系，以及我们自己的技术项目应用到产品线里的价值评估情况。

- **Web富应用**，有编辑器，个性化的可视化展现，百度脑图，可视化的矢量公式编辑等。

- **还有就是全端技术**，比如WebUploader的快速上传组件、移动端的组件库GMU和之前在做的轻应用中的轻组件。

**轻组件是怎么回事?**

**刘平川**：原来的组件要么是原生客户端的一套组件，要么是Web的一套组件，我们认为轻组件就是这两套的综合。比 如一个滑动的图像展示，如果你要在目前的Web上展示的话，它的流畅度不高，用原生的话就会很快。所以如果你用JS在我们提供的API上写代码，它会在不 同的运行情况下用不同的方案。这就像PC时代Flash插件的思路，没有Flash插件就变成最原始的显示，有Flash运行时体验会更好。这就是一种轻 组件的思维。

**为什么你们需要的人是全栈工程师?**

**刘平川**：我们在FEX博客上招聘要求全栈/全端工程师，还要求充分的自驱。在公司内部我们团队会维护自己的服务器和知识库的系统与机器。这些搭建都是我们自己完成的。虽然前端数据系统的Hadoop不是我们自己搭的，但是MapReduce的脚本都要自己写。

因为我们的工作不是狭隘的前端领域，也不局限于语言，需要涉及前后端。Web服务器以前就叫前端，开发流程优化这些工作我们都做。而流程的优化像持续集成与前端的运维就涉及到传统的前端和后端，这就要求我们在技术上也要掌握这个方向。

**基础团队对于整个公司的价值是什么?**

**刘平川**：我们现在想得比较清楚，是因为也经历过迷茫。去年各方向的负责人曾坐在一起讨论过我们应该做什么业务，我们做的业务应该有什么样的特性，我们应该怎么样跟产品线合作，整个前端和后端怎么合作。讨论过后我们发现，基础团队做的事应该是业务团队“重要而不紧急”的事。

我们认为对于业务团队来说，基础技术可要，也可不要。没有我们业务团队也能招人做，但业务团队需要我们，是因为有我们他们能做得更快更好

更专业。如 果业务团队使用了你的技术，那么基础技术团队的水平就决定了业务团队开发的技术架构的平均水平，这体现在开发效率、开发质量、带宽利用与人力成本上。

**带领FEX团队的过程中，你有什么感触?**

**刘平川**：很多的触动都是这个团队的同学给我的。我认为他们很多人比我优秀，我在团队中的作用就是让大家同心协力，有一个目标感，一起把一个事情做好。

比如团队的多益，技术非常好，之前我一直觉得应当给他技术上挑战很大的事情去做。后来他主动去搭团队的blog，把原来分散的代码仓库转移过来。而 且他在博客上也发表了很多文章，也主动写了我们的开源面试题。我问他，你会不会期望做更多技术上有挑战的事，他回答说，他很希望能做这些帮团队扩展技术影 响力的事情。很多技术能人可能比较喜欢独善其身，不愿意把个人与团队放在一起。我很幸运的是团队里都是既追求技术理想，也愿意配合团队目标的能人。团队有 共同的价值观，每位工程师都希望把各自方向的技术做到极致。

百度最佳员工我们部门就有5个。我们还拿到公司级别的 2个平台工具奖，1个最高创新奖。其实大家心里面都希望未来能拿一个公司的最佳团队奖。我觉得既然大家都能这样想，那我也不怕有什么事情做不成了。

# 数据库的最简单实现

作者：阮一峰

所有应用软件之中，数据库可能是最复杂的。

MySQL的手册有3000多页，PostgreSQL的手册有2000多页，Oracle的手册更是比它们相加还要厚。

但是，自己写一个最简单的数据库，做起来并不难。Reddit上面有一个帖子，只用了几百个字，就把原理讲清楚了。下面是我根据这个帖子整理的内容。

## 一、数据以文本形式保存

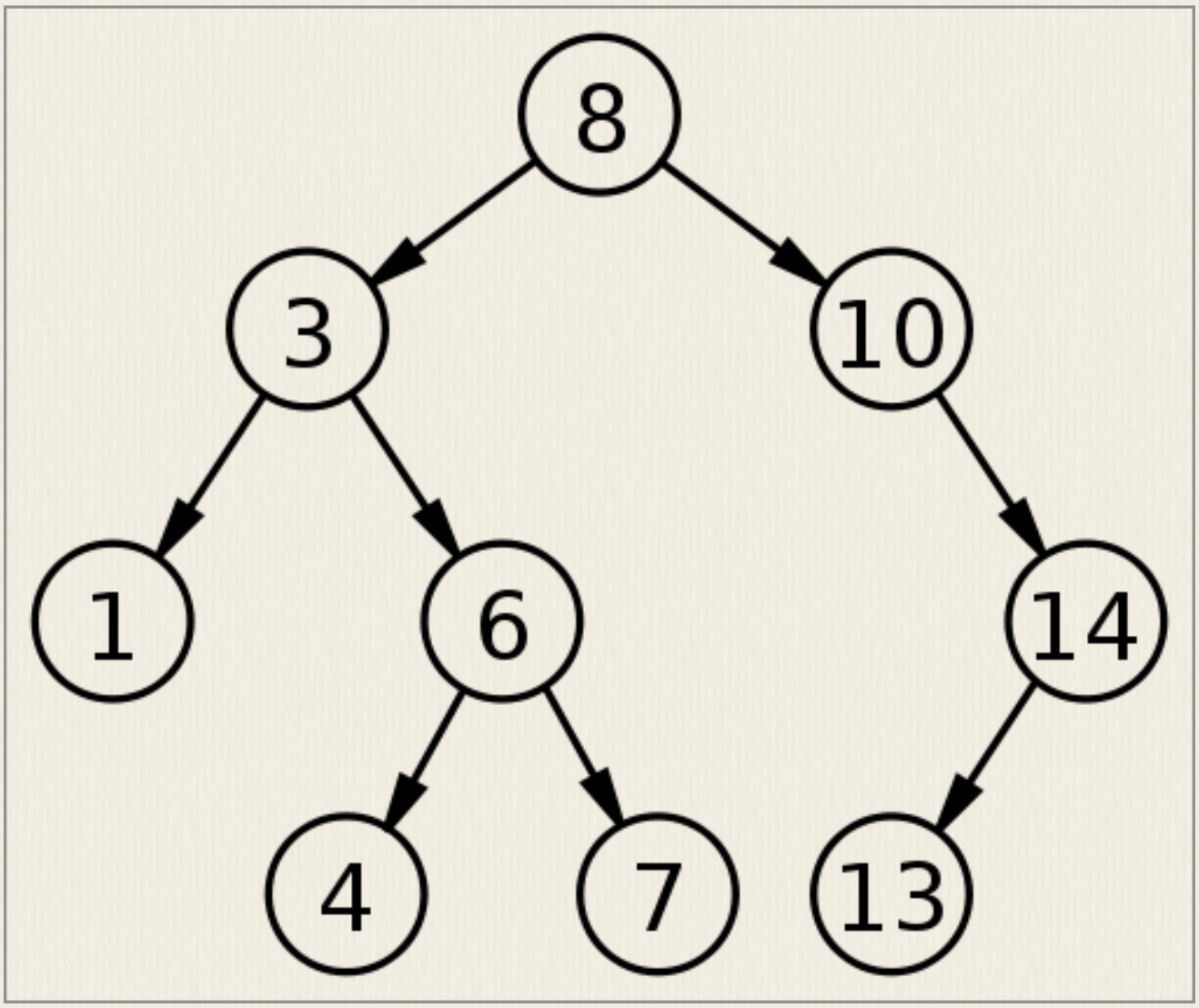
第一步，就是将所要保存的数据，写入文本文件。这个文本文件就是你的数据库。

为了方便读取，数据必须分成记录，每一条记录的长度规定为等长。比如，假定每条记录的长度是800字节，那么第5条记录的开始位置就在3200字节。

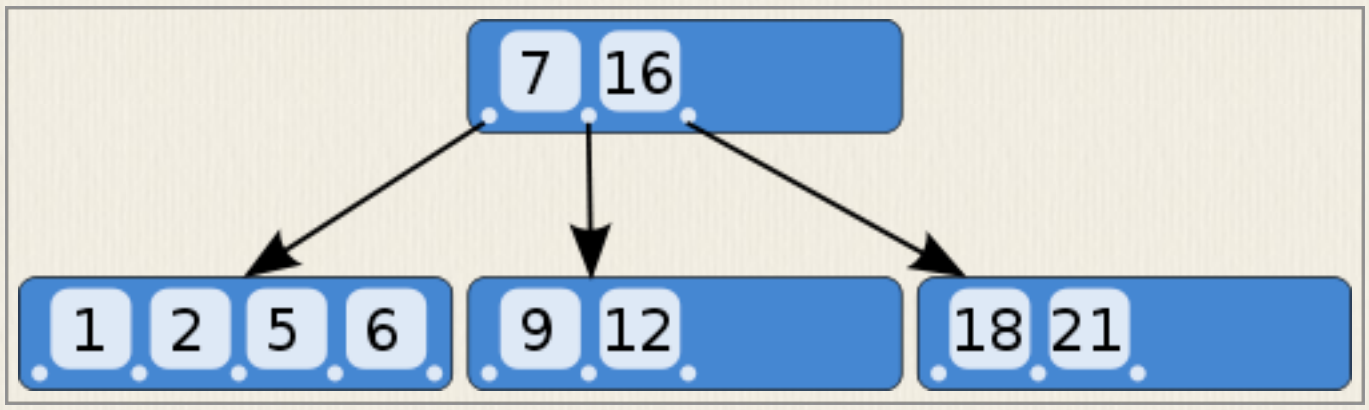
大多数时候，我们不知道某一条记录在第几个位置，只知道主键（primary key）的值。这时为了读取数据，可以一条条比对记录。但是这样做效率太低，实际应用中，数据库往往采用B树（B-tree）格式储存数据。

## 二、什么是B树?

要理解B树，必须从二叉查找树（Binary search



tree）讲起。




二叉查找树是一种查找效率非常高的数据结构，它有三个特点。

（1）每个节点最多只有两个子树。

（2）左子树都为小于父节点的值，右子树都为大于父节点的值。

（3）在n个节点中找到目标值，一般只需要log(n)次比较。

二叉查找树的结构不适合数据库，因为它的查找效率与层数相关。越处在下层的数据，就需要越多次比较。极端情况下，n个数据需要n次比较才能找到目标 值。对于数据库来说，每进入一层，就要从硬盘读取一次数据，这非常致命，因为硬盘的读取时间远远大于数据处理时间，数据库读取硬盘的次数越少越好。

B树是对二叉查找树的改进。它的设计思想是，将相关数据尽量集中在一起，以便一次读取多个数据，减少硬盘操作次数。

B树的特点也有三个。

（1）一个节点可以容纳多个值。比如上图中，最多的一个节点容纳了4个值。

（2）除非数据已经填满，否则不会增加新的层。也就是说，B树追求"层"越少越好。

（3）子节点中的值，与父节点中的值，有严格的大小对应关系。一般来说，如果父节点有a个值，那么就有a+1个子节点。比如上图中，父节点有两个值 （7和16），就对应三个子节点，第一个子节点都是小于7的值，最后一个子节点都是大于16的值，中间的子节点就是7和16之间的值。

这种数据结构，非常有利于减少读取硬盘的次数。假定一个节点可以容纳100个值，那么3层的B树可以容纳100万个数据，如果换成二叉查找树，则需 要20层！假定操作系统一次读取一个节点，并且根节点保留在内存中，那么B树在100万个数据中查找目标值，只需要读取两次硬盘。

## 三、索引

数据库以B树格式储存，只解决了按照"主键"查找数据的问题。如果想查找其他字段，就需要建立索引（index）。

所谓索引，就是以某个字段为关键字的B树文件。假定有一张"雇员表"，包含了员工号（主键）和姓名两个字段。可以对姓名建立索引文件，该文件以B树 格式对姓名进行储存，每个姓名后面是其在数据库中的位置（即第几条记录）。查找姓名的时候，先从索引中找到对应第几条记录，然后再从表格中读取。

这种索引查找方法，叫做"索引顺序存取方法"（Indexed Sequential Access Method），缩写为ISAM。它已经有多种实现（比如C-ISAM库和D-ISAM库），只要使用这些代码库，就能自己写一个最简单的数据库。

## 四、高级功能

部署了最基本的数据存取（包括索引）以后，还可以实现一些高级功能。

**（1）SQL语言是数据库通用操作语言，所以需要一个SQL解析器，将SQL命令解析为对应的ISAM操作。**

**（2）数据库连接（join）是指数据库的两张表通过"外键"，建立连接关系。你需要对这种操作进行优化。**

**（3）数据库事务（transaction）是指批量进行一系列数据库操作，只要有一步不成功，整个操作都不成功。所以需要有一个"操作日志"，以便失败时对操作进行回滚。**

**（4）备份机制：保存数据库的副本。**

**（5）远程操作：使得用户可以在不同的机器上，通过TCP/IP协议操作数据库。**

（完）

原文链接:
http://www.ruanyifeng.com/blog/2014/07/database_implementation.html

# TokuMX使用小计

作者:Yurii

最近因为工作的缘故，接触了TokuMX，尝试下来感觉不错，值得介绍给大家。

事情的起因是要解决MongoDB的问题。系统中需要保存程序输出的运行信息，这类信息比程序语言的log更高级，但又不如业务操作日志高级，是某 些时候发现问题的关键证据，所以必须保存。因为格式不太规范，又需要方便检索，所以文档型NoSQL的MongoDB是比较好的选择。

但是，选择MongoDB就必然会面对磁盘空间的问题。我们的数据大概是这样的：每天的数据量不到200万条，单条数据的平均大小不超过4k，但MongoDB存一个月的数据就消耗了接近40G，最近三个月的数据则需要接近100G。限于具体的硬件环境，只能保存最近三个月的数据，但这无法满足业 务需求，所以必须另想办法。

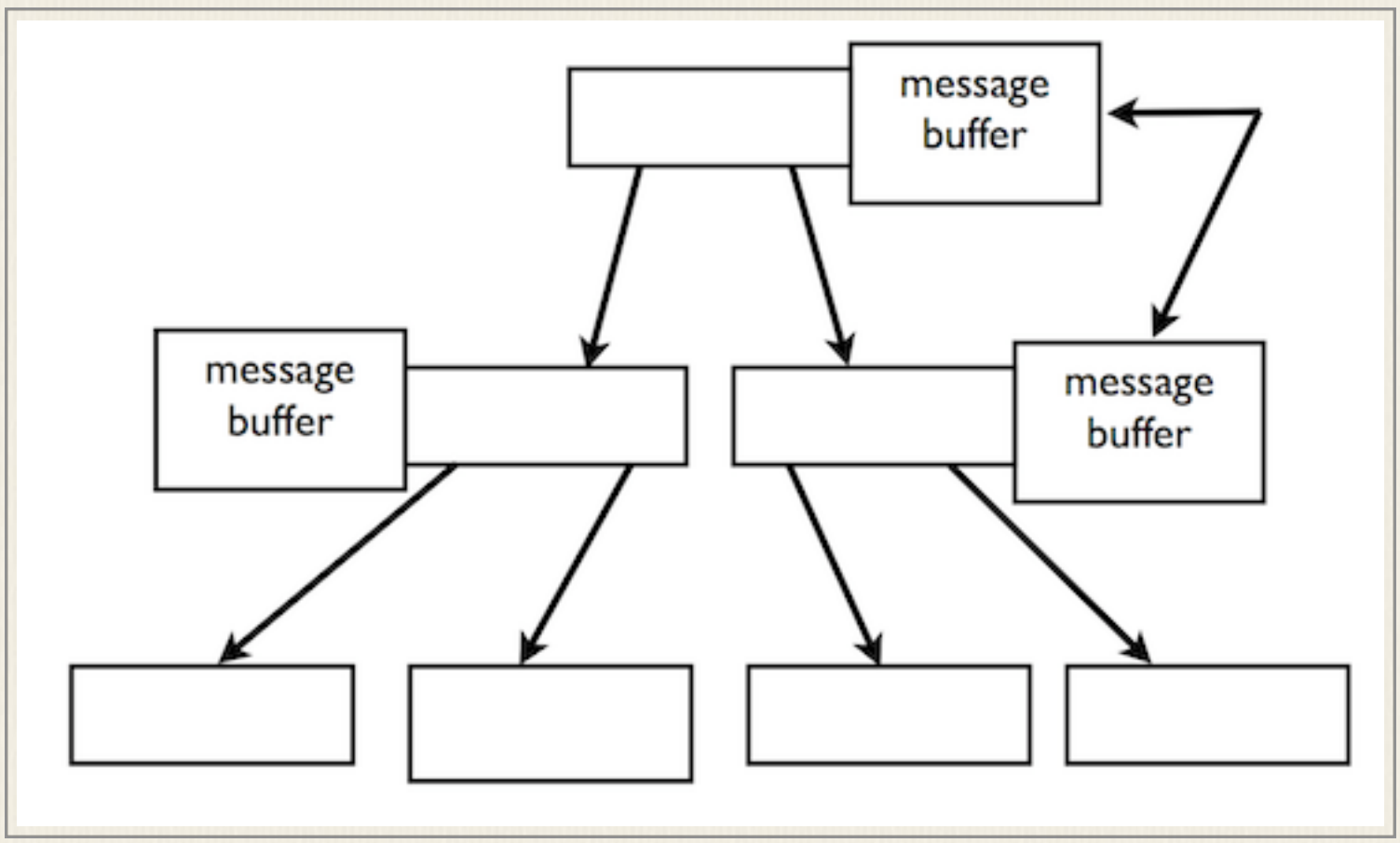
最终我们选定的方案是TokuMX。它是一款开源的、高性能的MongoDB发布（distribution），在提供与MongoDB完全兼容的客户端、API的同时，号称可以减少90%的存储空间，同时提供20倍的性能提升。我也了解到，已经有一些生产系统在使用TokuMX，反馈不错（比如 这里 和 这里）。

经过我的测试，从MongoDB迁移到TokuMX非常简单：用mongodump将原有数据导出，再在安装了TokuMX的机器上 mongorestore即可。原先用MongoDB需要102G的数据，采用默认的zlib压缩方式导入TokuMX之后，只有2.2G，同时导入速度 大大提高（至少有10倍的提高），而查询性能没有

降低（QPS在2位数左右，使用索引）。这个对比是我不敢想像的，它直接解决了现在的问题。

对着这份数据，我不免好奇TokuMX究竟使用了怎样的技术？就我现在的了解，减少磁盘空间占用主要是在存储层使用了压缩方式（TokuMX宣称，如果不使用压缩，TokuMX的磁盘占用也比MongoDB少10%左右）。这种思路不稀奇，5.x版本的MySQL中，如果设定file_format 为Barracuda，也可以直接对表做压缩，同时外部操作不需要做任何变化。TokuMX的提高写入速度则相当有趣，按照TokuMX的做法是使用分形树索引（Fractal Tree Index），替代了所谓“已经有40年历史的B树索引”，按照Wiki上的说法，TokuMX是分形树索引进行商业应用的典型。

“分形”是一个数学上的概念，大略来说，指的是“事物的每一部分都近似整体缩小后的形状”。TokuMX的分形树索引，严格说起来更像“B树 + 批量写入”，与B树的不同在于，分形树的每个内部节点都带有自己的缓冲区，它存储尚未落实（pending）到叶子节点的数据，默认情况下写入只会到缓冲 区，缓冲区填满之后会把所有的写操作刷（flush）下去。



我顺手翻译了TokuMX的一篇介绍文章，供大家参考。

http://www.luanxiang.org/blog/archives/1760.html

再附两份参考资料

percona的TokuDB和TokuMX介绍文档
http://www.percona.com/live/london-2013/sessions/fractal-tree-indexes-theory-practice

Facebook的人做的性能对比评测
http://smalldatum.blogspot.com/

推特上的 @BohuTang 应该是 TokuTek 的贡献者，人非常好，大家有问题也可以和他讨论。

原文链接: http://www.luanxiang.org/blog/archives/1763.html

# 怎么评价淘宝 UED 的 Midway Framework 前后端分离?

作者: 知乎匿名用户

利益相关，我是一个无名的阿里基层小技术主管，必须匿了。

经过ebay 等海外风的吹拂，前后端分离成了阿里集团前端组织今年内部推行的几大领域之一。

目 前在公司内部还只是处在早期探索阶段。诚然经过半年的宣传和鼓吹，在前端工程师的圈子里已有了不小的影响力，但目前还缺乏后端工程师的理解和参与，也没有 重量级的架构师（阿里架构师一般是指后端架构师）和技术管理层的充分讨论和鲜明支持，目前还是前端们的自娱自乐居多，各部门独立玩的居多，但已经呈现出比 较活跃的发展态势，也是内部技术讨论热点，可谓道路曲折，前景光明。midway只是一个方案，不代表阿里在这方面的全部成果，很早之前就有 nodejs/
前后端分离的相关技术产品出世了。

在此不想评价midway 的技术水平，应该说这是一个从实际场景出发、面向工程的技术架构，还在 探索。从纯技术意义上来说，其实并没有非用不可的理由，midway宣称的那些好处，没有哪一条完全站得住脚，包括最基础的前后端代码共用和模板共用，仔 细研讨起来也是过于理想主义的，忽视了实际工程的困难；至于利于性能优化或者bigpipe更是贻笑大方。

如果说前后端分离有什么好处，我觉得有三，需要辩证地看:

**1，扩大了前端的势力范围，**

对 技术影响力、招聘、造轮子、发paper、晋升答辩极为重要。要知道在阿里前端最高的级别是P8，人数则一只手能数的过来。这不能不说是前

端这一层过于 薄、离业务核心太远导致的，现在从技术上侵入服务端，技能树终于开了新分支，可能性猛然大了无数倍，一下子似乎发现了新大陆。对个人成长的渴望，是大量面 临瓶颈的资深前端工程师无法抗拒的诱惑！

**2，前端开发模式的变革。**

自己玩后端，技术上、项目上都不用去看后端项目的排期和配合了，极大提高了开发和测试的便利度、自主性。阿里很早就了nodejs写的开发服务器用于内部测试，现在走到线上，也是厚积薄发，这为提高开发效率，进一步榨取前端劳力提供了可能。

**3，促进阿里后端web框架的进化。**

这 一点是没有很多人甚至阿里人没注意到的（前端不懂webx，后端重心后移），webx框架发布已经很久，近期的技术升级非常少，技术形态已经陷入僵化，对 现在的移动互联网趋势没有敏锐的发现机会并适应变革，很遗憾地缺席了新时代。nodejs来了，洗牌开始，前端人员天然对多终端、多屏、交互、体验的关注 会为后端框架吹过一缕新风，给这停滞的技术领域注入活力。——所以我是赞成后端工程师学习nodejs的。

同时问题也很多，这些困难和隐患不能不提：

**1，不可忽视的后端技术门槛。**

不 能不提这茬，即便midway很务实的只挑选了视图层作为主攻方向，但不可避免的控制器层还会带进来大量技术问题需要解决，比如配管、部署、日志监控、运 维工具、SOA、加解密、事务、缓存策略、消息队列、异步调用、安全问题，总有避不开的暗礁。对阿里目前的后端技术栈来说，这些技术背后是无数的系统和平 台，缺一不可，midway目前还是个玩具。更何况nodejs/web framework本身都在飞速演变之中，ES5到ES6，技术特性变化剧烈。即便由前期探路者完成了基础设施建设，后期他人进入的学习成本也非常高。时 间长了，前端工程师本身又自然会分化成纯前端和nodejs工程师，如此前后端天然又产生了隔阂，呵呵，分久必合，合久必分。

**2，前端不务正业。**

正是因为困难重重，后端技术比前端复杂，加上新领域容易出成果，客观上造成了前端精英力量会大量倾斜到这个领域，进而使纯前端技术停滞不前，内部人心浮动，如果管理上不能很好理清长期目标和短期计划、前后端如何分离等关系和利益的要害问题，对公司来说是喜忧参半的。

**3，JS的技术缺陷**

因 为ES一直在进化中，ES6的yield部分解决了callback hell，module也初步有了，但是楼上推崇的前后端代码共享和模板共享，在我看来恰恰不太可行。对阿里来说，后端代码是需要保护的，大量模板、校验 函数等代码如果暴露出去，对网站的安全是一大威胁，这个好处实在鸡肋。譬如最近我们就发现JDK的一个加密类库在js中根本找不到替代方案，自己写的话实 在力有不逮。nodejs还太幼小，还有很多技术问题就不展开了。

**4，好处不够明显，推广困难。**

这才是 midway真正的死穴。如前文所说，目前的前后端分离运动还是部分前端工程师的孤立行为，绝大多数后端工程师和一线主管对此毫不关心（我除外），因为 midway宣称的痛点不是他们的痛点，宣称的优点只有节约后端人力这一条对他们有点意义，而这些人才是决定midway命运的人！随着时间的推移，这可 能缓解，也可能尖锐，毕竟这是前端们的西部大开发运动，是扩地之战。

**总结。**

目前midway至少还有半年的路要走，前端要克制自己的冲动，有限度使用nodejs，相信在这个领域大有可为，但那时候他们就不是前端了，他们会有一个响亮的名字叫 “NodeJS工程师”。

#_BEGIN

7月5日 12点 ，看到小卒的回答后我觉得有必要补充一下：

@小卒， 也许你觉得我在抹黑你，但我绝未针对midway，我也不认识这个团队的任何人，我描述的是管理上的大势，考虑引入一种新的后端语言对技术部门的长远影 响，不针对具体case和主观动机，只评估客观结果；另外，不想当将军的士兵不是好士兵。君子言利，才是社会进步的不竭动力吧！:)

至于我自己，一直是nodejs的拥护者，早在2011年就写nodejs了，可能比这里的很多人都要早，人活得长了点，也创过业，不小心成了一个老前端+老后端+半个运维+半个产品经理，不谦虚地说能写半打语言，我肯定打死也不会承认自己有编程语言偏见的，呵呵，但语言的差异和强弱是客观事实，承认一 门语言的缺陷并不会妨碍我们对它的使用，是不是?

后端技术门槛的存在，这一层认知障碍对你来说是很难感受到的，因为你有后端背景，但不要忘了前端同学们的普遍情况，有多少人有后端开发经验呢？我列出那些 技术名词，不是强调其技术实现（很多中间件客户端已经被数据平台部实现了）的难度，而是强调前端同学掌握这些技术领域的成本，开发、调试、摸出最佳实践， 这绝对是新的东西。现在前后端分离是在淘宝首页这种业务逻辑很少的场景中测试的，重点在view层，controller层可能只是简单的data collector，不知道你们考虑过复杂业务和交互场景下如何使用没有？那时候controller层会非常厚重，它不仅仅是取数和渲染视图，还会写入 数据、还有事务、队列和锁等杂七杂八，还有更复杂的业务逻辑需要在这一层进行，把一切复杂性委托给后端接口是不够的，因为再牛的服务化也不可能给你提供所 有粒度的封装，到时候node必然需要侵入到业务细节。代码复杂度一上升，callback hell谁也hold不住。

这一切对前端的挑战还不够大？也许你们可以试试怎么把下单系统给nodejs化。不客气的说，很多前端工程师一直在一维空间里存活，是单线程思维模式，对 并发、事务、一致性、分布式等问题的理解基本空白，技术场景后移以后，要学习的东西真的很多很多！ 所以我的意见是，如果前后端真的分离了，最后必然出现 NodeJS工程师，以区分纯前端工程师，不要奢望每个人都是大牛，这不科学。

前后端分离问题，其实是富客户端化大潮中的一个浪花，本质上还是移动时代对MVC模式中VC前移的内在呼唤。大家都希望后端老老实实只提供M就够了，别管 其他，View和Controller要么在app里，要么在H5中，实在不行啊，也得在nodejs里，后端该干嘛干嘛去，呵呵。说到底是在革Java Web Frame-
work们的命，这也是我的担忧，所以后端工程师的理解与支持才那么重要！

最重要的是，我花了很多业余时间观察集团内nodejs的发展，基本每个项目源码我都翻看过，我还特意和数据平台部的一些同事聊过相关的进展，其实是在琢磨如何在我的团队里引入的，所以如果让你觉得我对xxx有不屑，恩，这不是真的。我还是很佩服有想法、干实事的人的，加油！

#_END

原文链接: http://www.zhihu.com/question/23512853/answer/27533646

# 游戏软件开发，硬件的架构影响有多大?

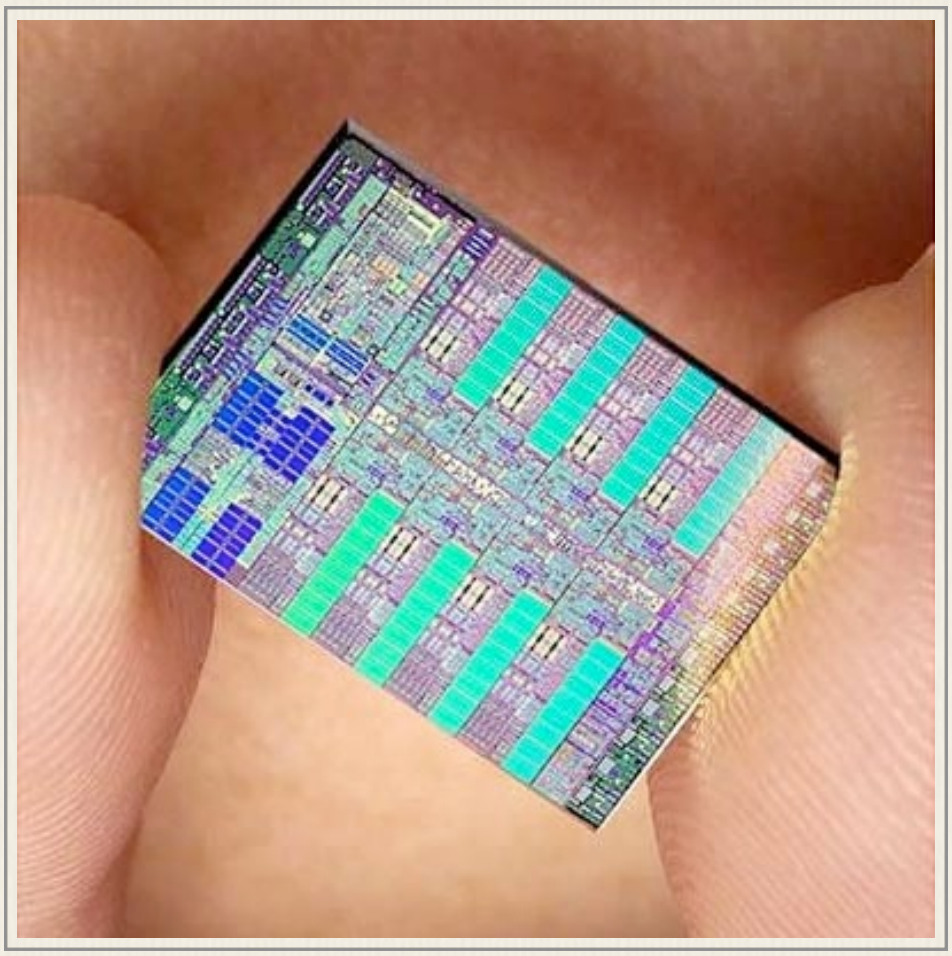
作者:Milo Yip

影响很大。硬件的功能及性能直接影响游戏软件的开发。

题主谈及PS3和Xbox360，本人参与过两个相关游戏项目的开发，就简单介绍一下它们硬件的差异如何影响软件开发。

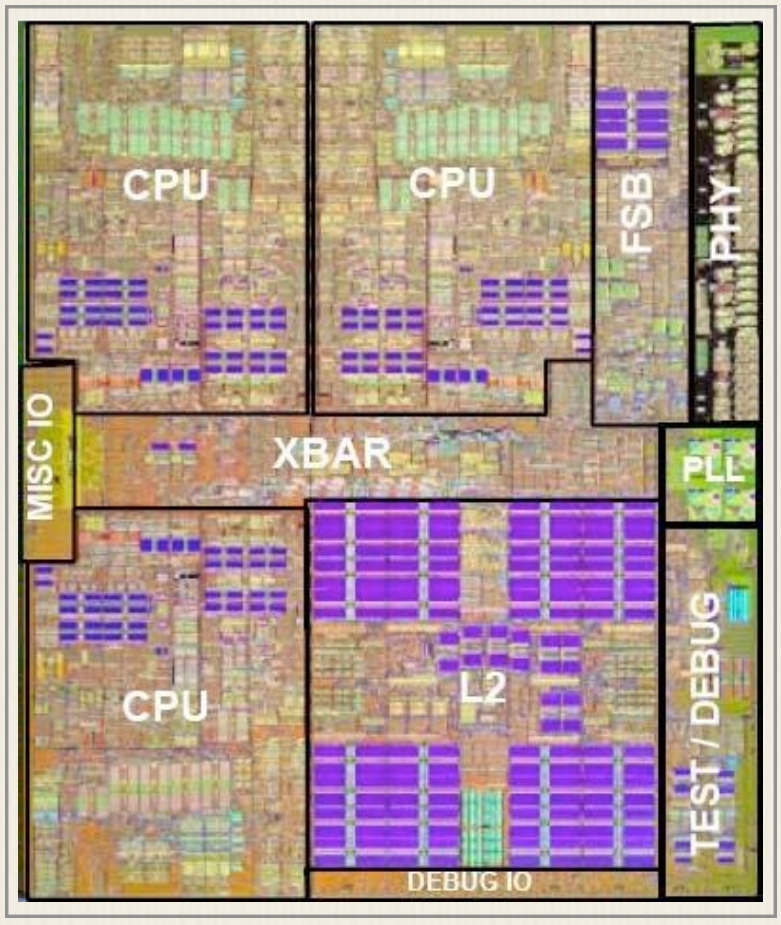
PS3 的特别之处，是它有一个主要的CPU（称为PPU），另外有6个可用的辅助处理单元（称为SPU）。软件开发难处在于，那些SPU各有256KB本地内 存，开发者要把想并行计算的数据从主内存打包发送到SPU，完成工作后再把结果抄回来！SPU还用另一套指令集、编译程序!

PS3 Cell处理器



而Xbox360则是3核CPU，能正常地访问512MB主内存（统一内存架构）。虽然不是x86架构，但能直接用正常的多线程方式来做并行。

Xbox360 Xenon处理器



至于SDK，微软提供Direct3D 9变种的API，移植PC的代码变得简单得多。而PS3的图形API则是非常低阶的，显存的地址分配都要开发者自己实现。另外，游戏的优化是需要针对个别CPU/GPU去做的，例如用该CPU/SPU的SIMD指令集去编程。

如 果同时还要跨平台至Wii，那才是恶梦！因为Wii的"GPU"是固定管道的！Shader Model 1.0都没有啊！做个材质效果要设置几十个渲染状态，像IQ题一样！想用cube map做反射效果要先把它贴在球体上渲染至frame buffer（render target都没有！），然后抄到纹理，再把纹理当作sphere map来用啊！！Wii所有内存加起来100MB都没有，而且有分快主内存、慢主内存、帧缓冲内存、纹理内存啊！！你说怎么影响软件开发！！

吐完。

原文链接: http://www.zhihu.com/question/23233122/answer/24055440

# 面试总结

作者:sunchangming

最近我在找工作，面试了多家公司：百度、阿里、小米、美团、Yahoo、Symantec、Amazon。其中Amazon面的是供应链（被HR忽 悠的），fail了。其它拿到了offer，但是都有些不如意。很多公司给我的薪水和职级只相当于毕业1-2年的人的水平，而我已经毕业7年了，所以这些 公司的尽管给我发了of-
fer，在我看来他们不过是婉拒了我。下面开始吐槽面试经历。

我认为无论是哪个公司的社会面试，看重的主要是以下几点：编码能力、算法、概念知识、项目经验、教育背景。

**编码能力**：我自己对编码能力比较看重，为什么别人花一个小时就做完的东西，你要花一天才能写完还全是BUG？但 是我此番面的公司基本都不太考察这个。Amazon用它的online test狠狠打击了一下我的对自己编码能力的自信。我想以后找点类似的题多练练。顺便透露一下，amazon的online test是在https://www.hackerrank.com/这 个网站上进行的。阿里是唯一一家没有考我手写代码的公司。我和阿里的面试官谈过这个问题，他说阿里更看重人的综合素质，而百度有点拿着铁箍买鸡蛋的感觉。 阿里问我：“你说你编码能力好，怎么证明？”我回答不上来。 接着又问我："你工作中出过的最大的BUG是什么？"我又回答不上来。

**算法**：这次几家公司的面试中，算法的重要性远远低于我的想象。即便我面的是凤巢很核心的算法部门，也没有考我很 难的算法问题。我认为算法能力主要分为三点：分析、设计、实现。分析最基础的，我将会持续加大在此方向的锻炼，多读paper多做数学推导。可惜，算法分 析对面试或升职涨薪都完全无用。面试官太看重候选人是如何寻求解法、设计算法，思维方向以及速度。我觉得这是不对的，因为除非面试官的算法能力远远超过并且涵盖候选人所知，否则对方怎么想的他怎么看的出来。甚至，我给出了一

个正确的解答之后，因为不是标准答案，面试官甚至不能理解我是不是对的（要在10分 钟内完整理解一个算法的设计或代码实现确实不易）。像百度这样的公司，挑选面试官挺随意的，你想应聘T6的职位它却找一群T6来面你。相比而下，微软的面 试官挑选就很严格，有资格做面试官的人很少。

**概念知识**：《crack the coding interview》一书中说，面试官是为了考察你的capabilities而非knowledge，像设计模式这样的东西都是不该出现在面试中的，因 为太经验化。而实际上，面试往往是以“你知不知道XXX"这样的模式往下走的。就面试中所谈论的技术话题的广度来说，Yahoo是难的，有个面试官很喜欢 跟我聊新技术，比如actor、协程、lock-free等。阿里的技术面试虽然很短（最短），但是问了我很多很深的东西，比如java7的新数据结构、 Mutex内部的队列是如何实现的、JVM的intrinsic、openjdk里hotspot的代码、有没有看过JSR、intel CPU近几年的架构变迁等等。

**项目经验**：这个环节我很不喜欢。这完全就是在考察一个程序员的销售能力。面试官希望看到你过去的项目很有技术难 度，用户量、并发量很大，用到的新技术很多，他不在乎你是怎样用非技术手段解决技术问题的，他也不在乎你怎样以很小的成本、很初级的技术实现了很核心的业 务需求。做销售的核心是名校毕业，见客户前准备充分，把相关的技术名词背的滚瓜烂熟，介绍自己的产品技术时滔滔不绝，会察言观色。每个技术名词都是一块金 箔，贴的越多，你就赢了。

**Behavioral question:** 据说Amazon至少有30%的面试时间都是在考察这个。实际上据我的经历，确实是这样。

最后说自己的选择。

就工作内容来说，我最喜欢的是百度和Yahoo，是做搜索广告。两家的薪水也给的差不多，所以我一直在这两家之间犹豫。

就公司的未来发展来说，小米应该是最好的，Yahoo可能是最差的。

就面试体验来说，小米和yahoo最好。小米每个面试官都会先介绍自己叫什么名字，并且会花很多时间介绍他们的项目。小米号称采用Google风格的招聘，只要基础能力好，就愿意要，至于去哪个项目就再谈。小米和yahoo的hr在谈薪水的时候都很直接，不跟我绕弯子。smth上有人

（ppstay)说"雅虎北研是国内外企不多的良心企业"，就我接触来看，此言不差。

最终去哪还没定下来，等我签完劳动合同了再来更新此文。 ^_^

原文链接: https://www.sunchangming.com/blog/post/4629.html

# 专访QQ大数据团队，谈分布式计算系统开发

作者:仲浩

摘要：他们前身是QQ成立之初后台3个基础团队之一的QQ运营组，当下致力于腾讯内部的分析系统，在离线及交互式计算系统上积累了大量经验，更是面向应用的数据解决方案ADs的作者。

NoSQL是笔者最早接触大数据领域的相关知识，因此在大家都在畅谈Hadoop、Spark时，笔者仍然保留着NoSQL博文的阅读习惯。在偶尔阅读一篇Redis博文过程中，笔者发现了 jacksu的个人博客，并在其中发现了大量的分布式系统操作经验，从而通过他的引荐了解了QQ成立之初后台3个基础团队之一的QQ运营组，这里我们一起走进。

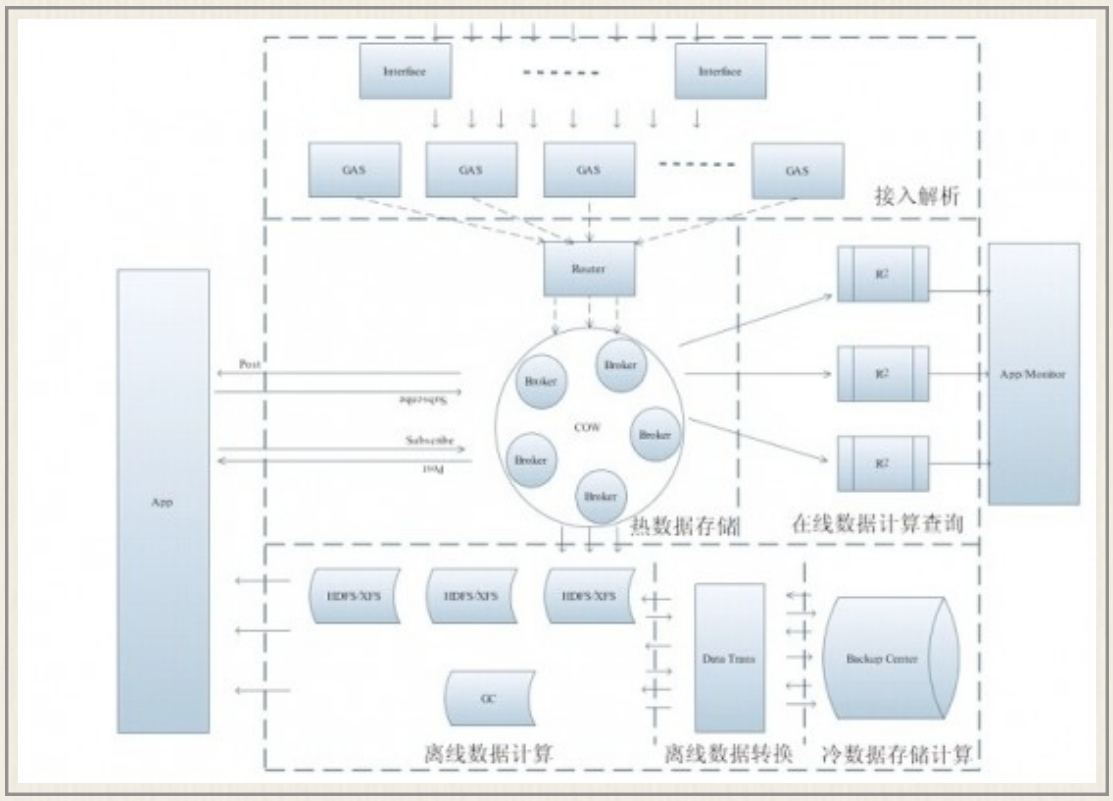

QQ大数据团队

**CSDN**：**首先，请介绍一下您的团队？**

**聂晶**：我们团队是社交网络事业群／社交网络运营部／数据中心／平台开发二组，前身是QQ成立之初后台3个基础团队之一的QQ 运营组。目前团队成员10人，主要负责社交网络事业群的基础数据挖掘系统和产品应用系统的研发和运营。作为腾讯内部较早研究并使用Hadoop的团队，结 合Hadoop、Spark等开源系统，推出面向应用的数据解决方案ADs（Aggregate Data services），涵盖数据整个生命周期；曾经面向复杂关系链计算，研发出圈子分布式计算系统等。目前，兴趣在于面向计算的分布式快速应用开发部署系统 ——R2，以及数据可视化的应用。

**CSDN**：**贵团队是ADs的作者，可否为我们介绍一下当下ADs在腾讯的使用程度，比如支撑的业务，处理的数据集，集群规模等。**

**聂晶**：ADs是腾讯即时通信线通用的，负责数据收集、分发的基础设施。ADs是一系列组件的统称，这些组件绝大多数为自主研 发，可以灵活组合起来支持实时和离线的多种数据需求。目前，ADs集群共700台各型服务器，日处理数据在2300亿左右，存储数据10PB+。为腾讯内 部5个部门，20多个业务线提供有效的支撑，比如数据查询、数据分析、产品统计、数据挖掘和用户推荐等。像QQ，手机QQ，以及其他通过即时通信工具接入 的业务，其基础数据都经由ADs对外提供服务。




图一ADs架构图

**CSDN**：众所周知，扩展性是大型网络架构中必不可少的一环，请结合腾讯的实践经验做一些**node rebalance**相关分享？

**聂晶**：扩展性，在我们看来，包含两种含义：第一种是功能的扩展性，还有一种是整个系统吞吐的扩展性。

对于功能的扩展，从系统层面上，可以做的是根据系统承载的功能，抽象成不同组件，不同组件之间的结合，可以灵活扩展出面对新场景的功能。比如，ADs就抽象出接入自动解析的GAS（General Analyses Service）组件，高吞吐存储的COW组件，数据转换的DataT组件。GAS＋COW就能提供应用的数据获取服务；GAS＋DataT提供给离线模型计算使用。

对于整个系统的吞吐扩展，一般都会设计成去中心化的结构，每个节点提供对等的服务能力。比如GAS就是如此，每个机器负责的是对等 的服务能力，如果机器死机或者阔扩容，通过配置中心更新节点路由，保证服务一致，加上一些消息探测的机制，即使在某些极端情况下没有更新路由，也不会丢失 消息。

**CSDN**：在线处理环节，你们自主研发了**R2**，可否分享一下与当下流行计算框架**Spark**及**Storm**的对比？

**聂晶**：首先，R2 跟已有开源项目最大的不同在于它从一开始就是为了面向实时服务而设计的，所以它对性能和低延迟和系统可用性要求更强，比如，在推荐好友业务中，需要在 200ms内返回数据，但是涉及处理的数据却可能高达几百MB，怎样提升计算降低延时，是一个挑战。其次，从架构上看，R2是一个对称的结构，没有单点。 节点可以做到即插即用，扩容缩容不影响服务，这对存在一定资源空闲的大型机房来说，可以随时使用空闲资源，节省成本。再次，从功能上讲，R2对一些特定的 迭代计算做了大量优化，使得很多智能算法的实现变得简单高效。

**CSDN**：在**ADs**中，你们使用**Hadoop**做离线处理，那么如此规模下，主要的挑战是什么，会遇到哪些坑，及需要避免的地方？

**聂晶**：

1. 目前前主要使用的还是1.0版本，由于1.0版本的单点问题，如果主控机器死机，对业务会造成较大的影响。

2. 对模型计算，涉及到大数据的频繁读写计算，效率着实不高。所以，对于此类业务，我们在逐步迁移到spark。

3. 多用户同时使用集群，千万要根据业务特性使用不同的调度器。

4. 在Hadoop自身文档还不够完善时，有些细节千万不能想当然，需要多试试。比如配置机器host时，hostname不能带下划线。

5. 千万不要让集群节点的磁盘容量差异太大，否则在大数据写入并且集群使用率较大时，容易出现写失败等问题。

**CSDN：在海量数据存储的过程中，在读写上是否遇到哪些问题？有没有调整系统默认的I/O调度策略或者是自己重写相应的文件系统？我说的是和Ext3/Ext2一个级别的文件系统。**

**聂晶**：默认机器一般是对硬盘做RAID5，但是RAID5相对于RAID0，写性能也是比较差，而且比较浪费空间 （Hadoop自己对数据有容灾），我们使用的磁盘都是RAID0。不同的调度器对性能影响很大，通过测试使用比较适合业务的调度器，SSD和机械硬盘的 差距就比较大，分别使用不同的调度策略。Ext3不同的日志级别对性能影响很大，建议关键业务进行性能测试，使用适合业务本身的日志级别。这里只是使用比 较成熟的调度策略，自己没有进行重写。

**CSDN：贵团队自主研发了数据解析服务GAS，可否为大家介绍一下主要特性？据悉即将开源？**

**聂晶**：GAS是一个通用的、实时的高性能数据解析框架，支持把不同格式的数据源，自动转换成一种格式，为后续组件提供无差别的流式数据服务。目前，GAS支持二进制协议、ProtoBuf协议、Json协议的解析。GAS的主要特点有：

- 吞吐量大，单机峰值可到10w+/s,可充分利用机器资源
- 提供通用的接口，方便扩展其他不同类型的协议
- 单个数据格式修改方便，实时修改，实时生效

GAS目前已经在公司内部开源，目前正积极准备对外开源的有关事项。

**CSDN：说到开源，可否透露一下腾讯当下使用的开源技术？都在系统中扮演着什么样的角色？顺便给大家谈谈使用开源技术的经验吧。**

**聂晶：**在两种情况下我们会使用开源技术：第一种情况，在较简单非关键的应用中有使用开源的技术，比如thrift， 我们在数据查询等一些小系统中有使用，开源技术的优点显而易见，可以节约开发成本，很容易的可以实现简单的需求。第二种情况，一些绕不过去的，比较成熟 的，会使用开源系统，比如Hadoop，Zookeeper。我们系统中，底层和关键模块都是自己开发，做到完全可控。

开源技术良莠不齐，一些冷门的或者不成熟的最好不碰。即使是成熟的开源技术，在使用中也是有各种坑。不过，成熟或者热门的技术，好处在于可以利用各种网络资源，也有成熟的社区，你遇到的问题，大部分别人也遇到过，容易解决。

**CSDN：无缝体验一直是服务交付中重要的一环，对于消除中间人，让实际使用者拥有一个更好的体验贵团队做了哪些努力?**

**聂晶：**ADs可以拿出说说。原来我们接入一个数据需要产品、开发、数据管理员多次沟通、多次联调以及多次数据质量确认，才可以完成一个数据的接入，效率极低。ADs出现之后，减少了数据管理员环节。产品通过ADs去管理、验收数据；开发根据产品的提单开发、自助测试，确认数据质量，知会产品 验收数据。


图二 数据接入图

原文链接：http://www.csdn.net/article/2014-07-03/2820520

# 前端之Android入门(6):屏幕适配

作者: HB仔

在前几篇文章，我们大致了解了Android开发的一些常规模式和方法，在这个过程中其实我们已经接触到了屏幕适配方面的相关知识，但是并没有深入地讲解这些概念。这篇文章我们将分三个方面进行详细讲解。

## 一，Android的度量单位

在Android中，定义组件尺寸的单位通常有dp和sp,那么它们跟我们经常使用的px有什么区别呢?

- px

px是Pixels的缩写，是常用的像素单位，对应的是屏幕上的点。移动设备的尺寸有很多种，它们的屏幕总像素也是不同的，使用px单位的组件尺寸会有不一致的情况。

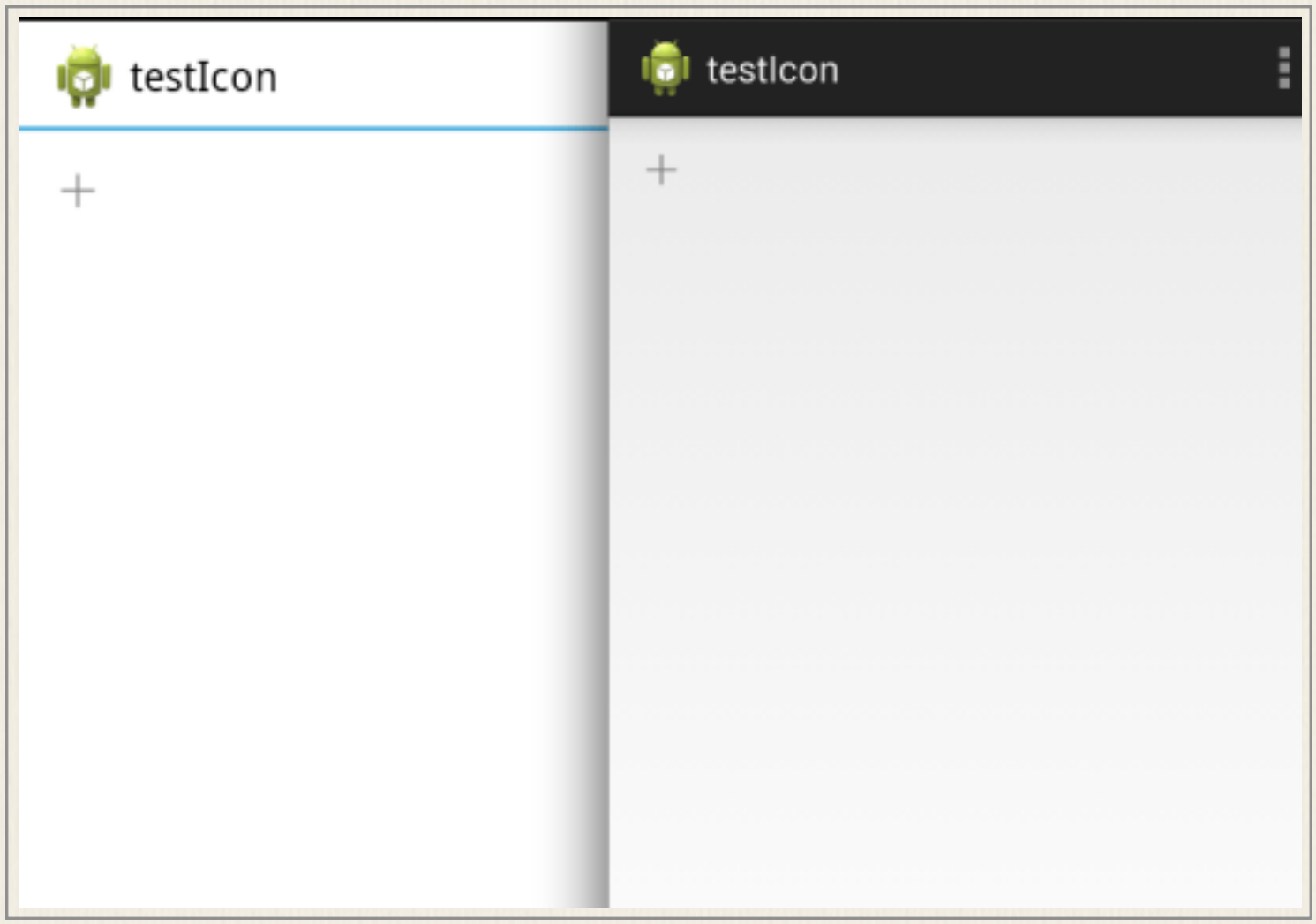
我们看个例子

可以看出这个加号icon在分辨率较高，尺寸较大的屏幕上反而显得略小

•dip/dp

dp(density-independent pixel)是与密度无关的像素单位，也就是dip。它是基于设备屏幕物理密度的抽象单位。1dp 表示屏幕DPI为160时1px的长度。DPI 越高的屏幕，屏幕绘制1dp 需要越多的像素，反之亦然。

我们可以将上个例子的图片宽高设置为30dp




可以看出两个加号icon的尺寸是一致的

•sp

sp(scale-independent pixel)是与缩放无关的像素单位。跟dp类似，唯一的不同是，该单位会受系统字体设置的影响，通常用在字体上。
Android在设计规范中规定了字体的常用大小:


使用dp/sp的好处是，无论屏幕DPI如何，组件总能表现一致。

## 二，Android的布局

在Android开发，为程序定义用户界面布局有以下方法：

- 在XML 文件定义
- 在程序代码中实例化布局
- 使用图形布局工具

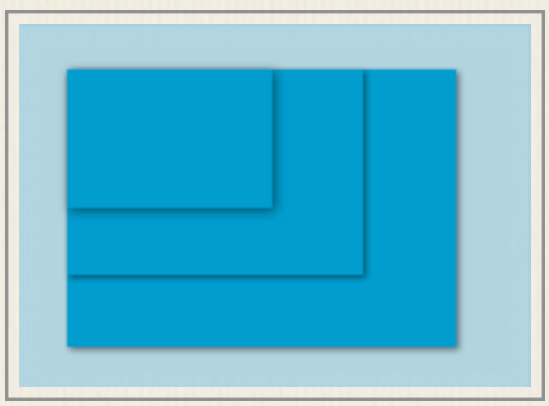
以上三种方法各有利弊，我们会根据需求场景来应用不同的布局方式，这里主要讲XML布局。

在XML文件定义组件，可以方便的管理程序的UI部分，并且分离表现和行为，使得代码容易管理阅读和管理，而且可以为不同的屏幕定制XML文件。

常用的布局有Frame Layout， Linear Layout， Relative Layout，List View 和 Grid View。

由于List View 和 Grid View 涉及数据部分，我们放到后面再讲，先从几个简单的开始 :)

Frame Layout



FrameLayout是最简单的布局，添加到这个布局的元素都是层叠的，当前元素会被下一个元素覆盖，相当于CSS中的绝对定位position:absolute。

从 Summary 的 XML Attributes 中可以知道这些属性的信息概要。

| XML Attributes | | |
|---|---|---|
| *Attribute Name* | *Related Method* | *Description* |
| android:foreground | setForeground(Drawable) | Defines the drawable to draw over the content. |
| android:foregroundGravity | setForegroundGravity(int) | Defines the gravity to apply to the foreground drawable. |
| android:measureAllChildren | setMeasureAllChildren(boolean) | Determines whether to measure all children or just those in the VISIBLE or INVISIBLE state when measuring. |

几个常用的属性:

1. android:foreground 设置布局的前景图，前景图不会被子元素覆盖
2. android:foregroundGravity 设置布局前景图的位置

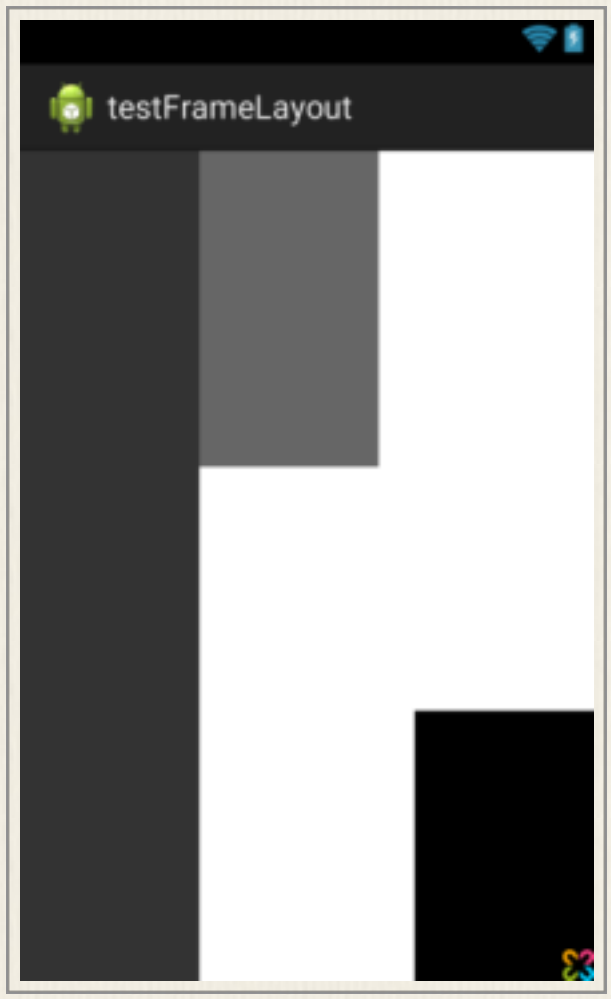
对于FrameLayout.LayoutParams，这里仅有android:layout_gravity属性，可以查看前面文章

http://isux.tencent.com/learn-android-from-zero-session3.html

我们可以实践一下:
新建一个Android项目，然后在lay-out文件夹找到布局xml文件并写入以下布局

```
1  <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
2      android:layout_width="match_parent"
3      android:layout_height="match_parent"
4      android:foreground="@drawable/logo"
5      android:foregroundGravity="right|bottom">
6
7      <View android:layout_gravity="bottom|right"
8          android:layout_width="100dp"
9          android:layout_height="150dp"
10         android:background="#ff000000"/>
11
12     <View android:layout_gravity="top|left"
13         android:layout_width="200dp"
14         android:layout_height="175dp"
15         android:background="#ff666666"/>
16
17     <View android:layout_gravity="top|left"
18         android:layout_width="100dp"
19         android:layout_height="match_parent"
20         android:background="#ff333333"/>
21
22 </FrameLayout>
```

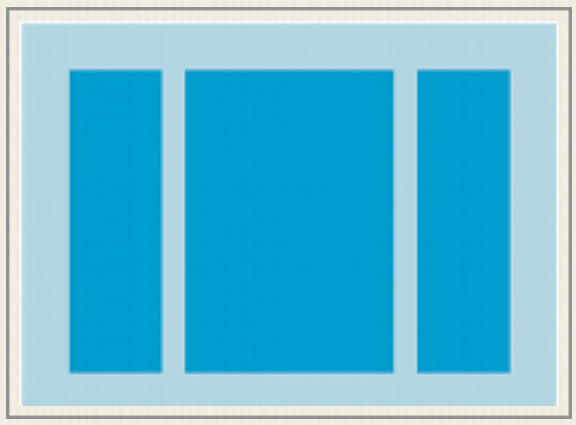
运行效果如下：




更多详细信息可以查看谷歌大神文档：

http://developer.android.com/reference/android/widget/FrameLayout.html

Linear Layout



线性布局在前面有所讲解，小伙伴们可以看这篇文章http://isux.tencent.com/learn-android-from-zero-session3.html

Relative Layout

相对布局可以指定元素的位置。元素的位置可定义为相邻元素的相对位置（例如元素A在元素B的左侧），也可以定义为父元素的相对位置（例如在父元素的中间或者左侧）。

使用相对布局可以方便的控制元素的位置。在没有定义时，所有的元素都是在父元素的左上方。

从 Summary 的 XML Attributes 中可以知道这些属性的信息概要。

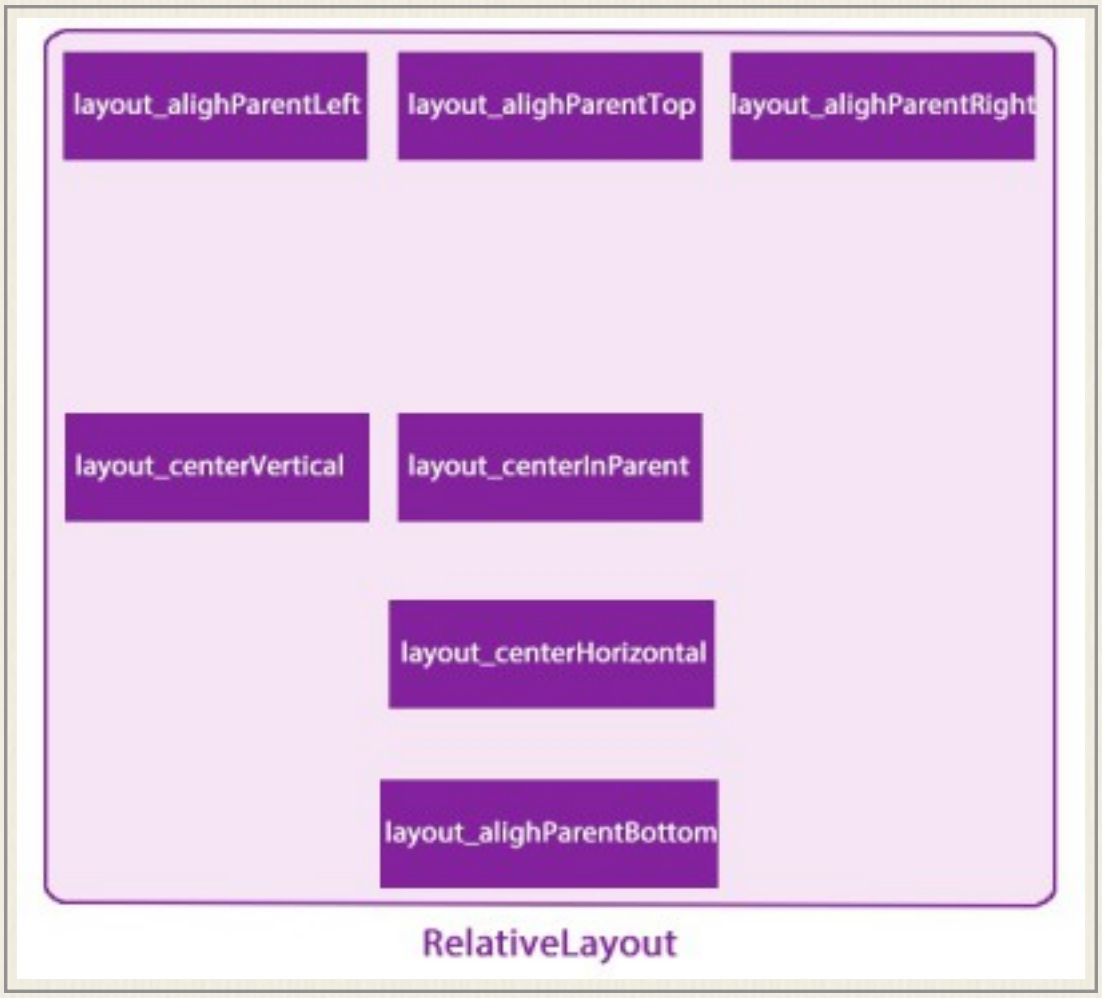
| XML Attributes | | |
|---|---|---|
| *Attribute Name* | *Related Method* | *Description* |
| android:gravity | setGravity(int) | Specifies how an object should position its content, on both the X and Y axes, within its own bounds. |
| android:ignoreGravity | setIgnoreGravity(int) | Indicates what view should not be affected by gravity. |

几个常用的属性:

1. android:gravity 设置其内容（文字、视图）在该元素内的位置
2. android:ignoreGravity 设置元素的ID使其不受gravity属性的影响

在 RelativeLayout.LayoutParams，有两类属性，一类是子元素在父元素的对齐方式，另一类是子元素相对其他元素的位置

下面是第一类属性的示意图




对于第二类属性，我们实践一下：

新建一个Android项目，然后在layout 文件夹找到布局xml文件并写入以下布局

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```

然后在values文件夹找到string.xml并增加以下代码

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">RelativeLayout</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="reminder">事件</string>
    <string name="done">确定</string>
</resources>

```

运行效果如下:


更多详细信息可以查看谷歌大神文档:

http://developer.android.com/reference/android/widget/RelativeLayout.html

## 三，Android的屏幕适配

1.屏幕相关概念

下面介绍几个常见的名词

- 屏幕大小

通过查阅设备信息可以知道设备的物理尺寸，也可以通过计算屏幕对角线的长度得到。
在Android的设计规范中，把屏幕分成了4类：Small，Normal，Large，Extra Large

- 屏幕密度

屏幕在物理尺寸范围内的像素数量。也通常指DPI(dots per inch)。屏幕密度越小，所包含的像素也就越少。
在Android的设计规范中，把屏幕密度划分为4类：Low，Medium，High，Extra High。

- 屏幕方向

屏幕方向有横向和纵向，这两种情况下的屏幕长宽比是不同的。

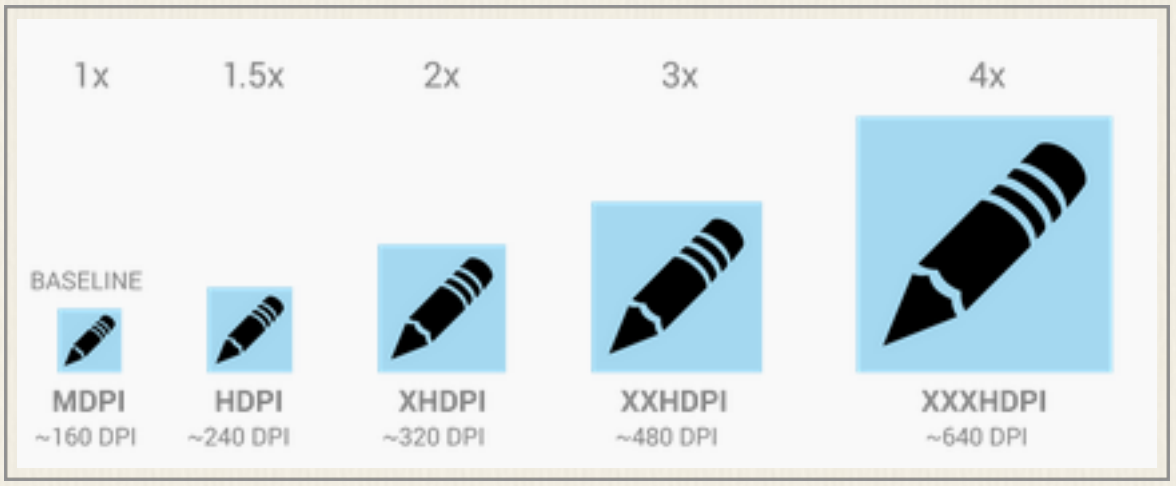
- 分辨率

跟电脑分辨率的概念类似，表示屏幕横纵方向的像素数，例如480*800。在为APP进行屏幕适配时，不能只考虑到像素，还有屏幕大小，屏幕密度等等。

- DPI

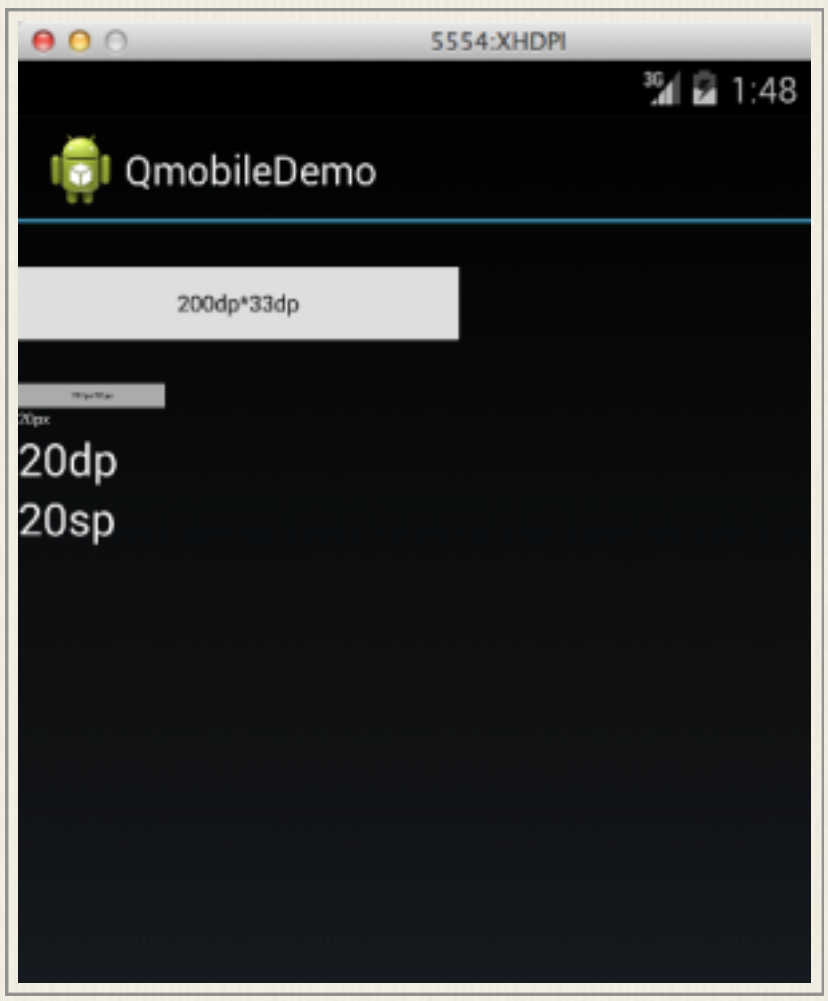
dots per inch，就是每英寸多少像素，通过下面公式可以得到。
DPI =（√（横向分辨率^2+纵向分辨率^2））/屏幕尺寸

在Android的设计规范中，DPI分成了5个档次：MDPI，HDPI，XHDPI，XXHDPI，XXXHDPI，它们的比例是 2:3:4:6:8

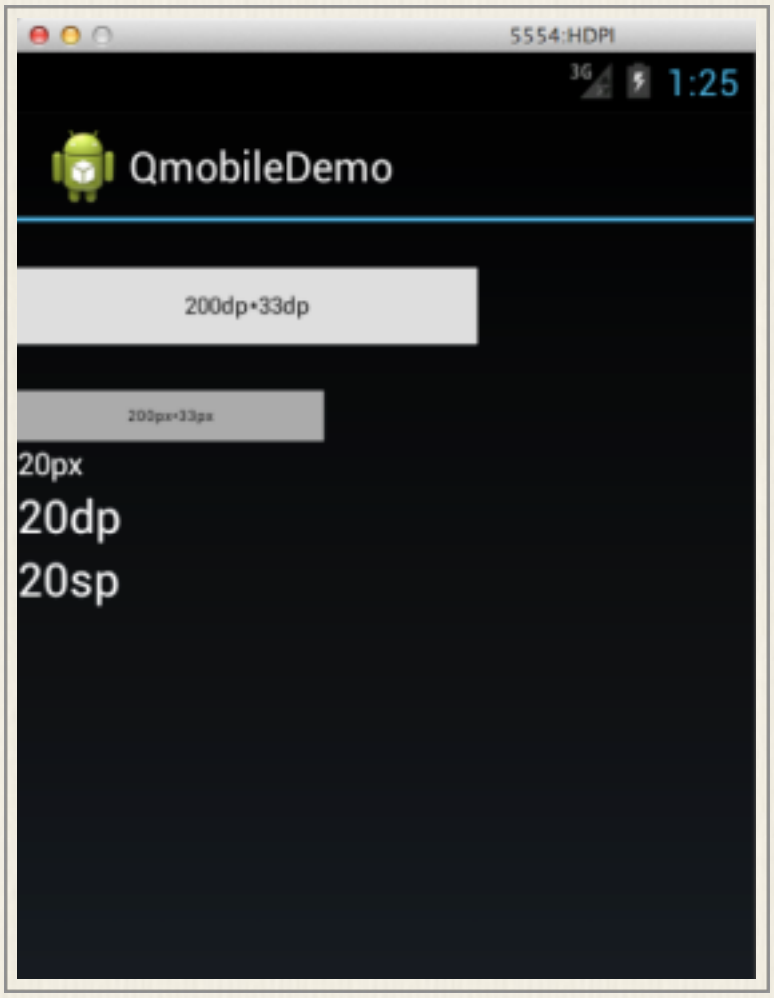



可以看一下不同的尺寸单位在不同屏幕的情况：

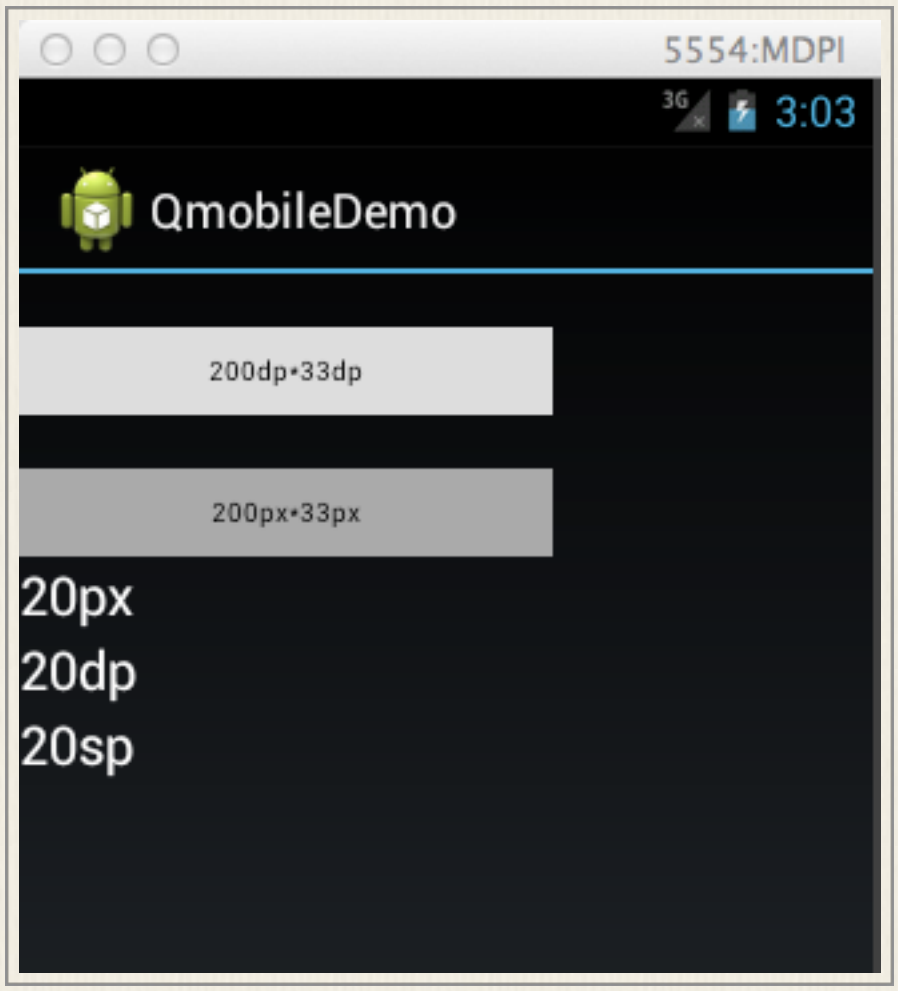
XHDPI屏幕(~320DPI)




HDPI屏幕(~240DPI)




MDPI屏幕(~160DPI)



通过以上可以得出:

- 在MDPI的屏幕(即160DPI)，1dp 和1sp 基本等于 1px 。
- dp 和 px的换算公式：px ＝ dp*(DPI/160)。
- sp 和 px的换算公式：px ＝ sp*(DPI/160)。

2.为不同的屏幕大小提供布局

通常来说，Android通过缩放使得APP可以适应屏幕大小，但是对于一些特殊情况，我们可以提供更佳的布局使得体验更好。例如，在大屏幕设备，可以通过调整UI组件的位置和大小使屏幕的空间可以得到充分利用。对于小设备，就只需要调整UI组件的大小就行。

布局常用的大小有small,normal,large和x-large，可以根据需要选择布局的类型进行配置。

常用的布局配置如下:

- res/layout/layout.xml
- res/layout-small/layout.xml
- res/layout-large/layout.xml
- res/layout-large-land/layout.xml

3.为不同的屏幕密度提供图片

对于android来说，可以使用的图片类型有.png，.jpg，.gif以及.9.png(点九图)。APP会为不同的屏幕放大或者缩小图片。 如果只为MDPI的设备提供了图片，那么对于HDPI的设备图片会放大，对于LDPI的设备图片会缩小。所以最好的方式是为不同的屏幕密度提供图片。

对于设计师来说，可以选择320DPI作为标准对APP进行设计，然后选择Photoshop/Illustrator这些工具，按照设备的比例， 导出不同尺寸的图片。例如，为标准尺寸设备(XHDPI)设计的200×200按钮图，就需要为HDPI设备导出150×150的尺寸，为MDPI设备导 出100×100的尺寸

然后将这些图片放到 res/

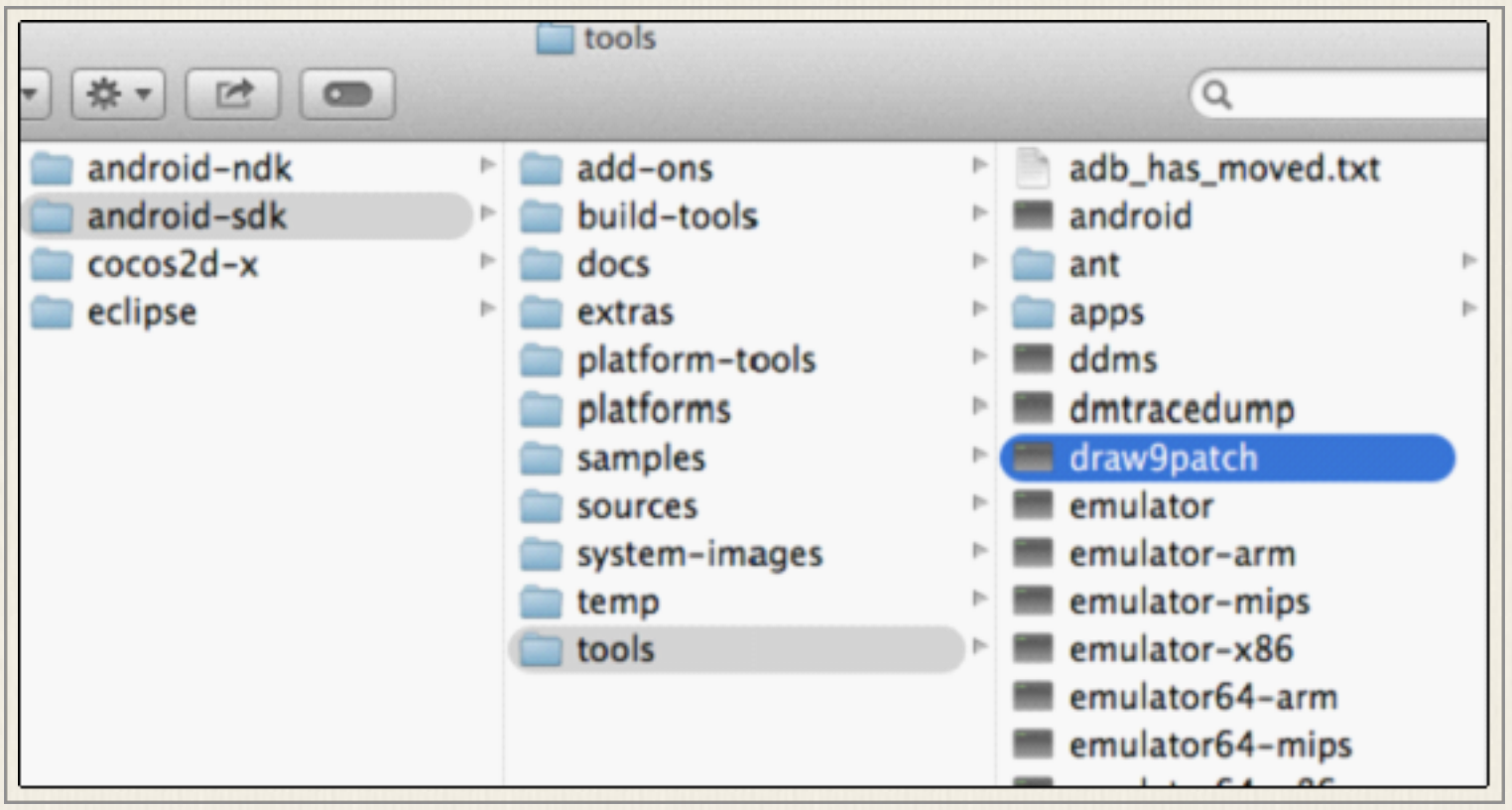
中对应的文件夹，APP会根据设备的情况选取图片。

设置如下：

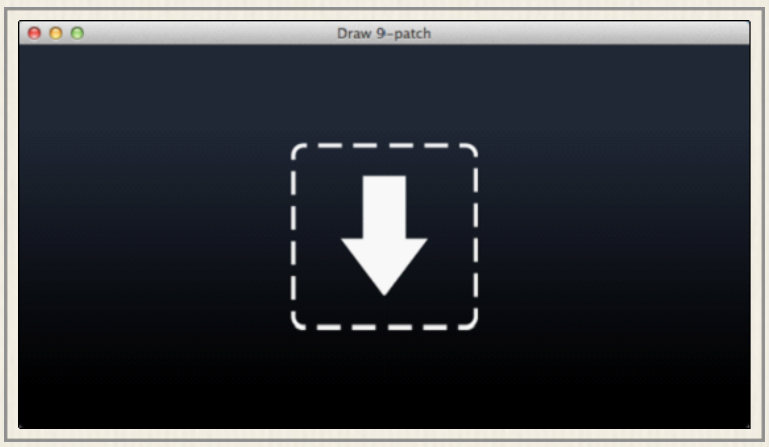
- res/drawable-xhdpi/image.png
- res/drawable-hdpi/image.png
- res/drawable-mdpi/image.png

4，点九图

“点九”是一种特殊的PNG格式，可以对图片指定拉伸区域和内容区域。在SDK里面的 tools/文件夹就能找到制作这种图片的工具。



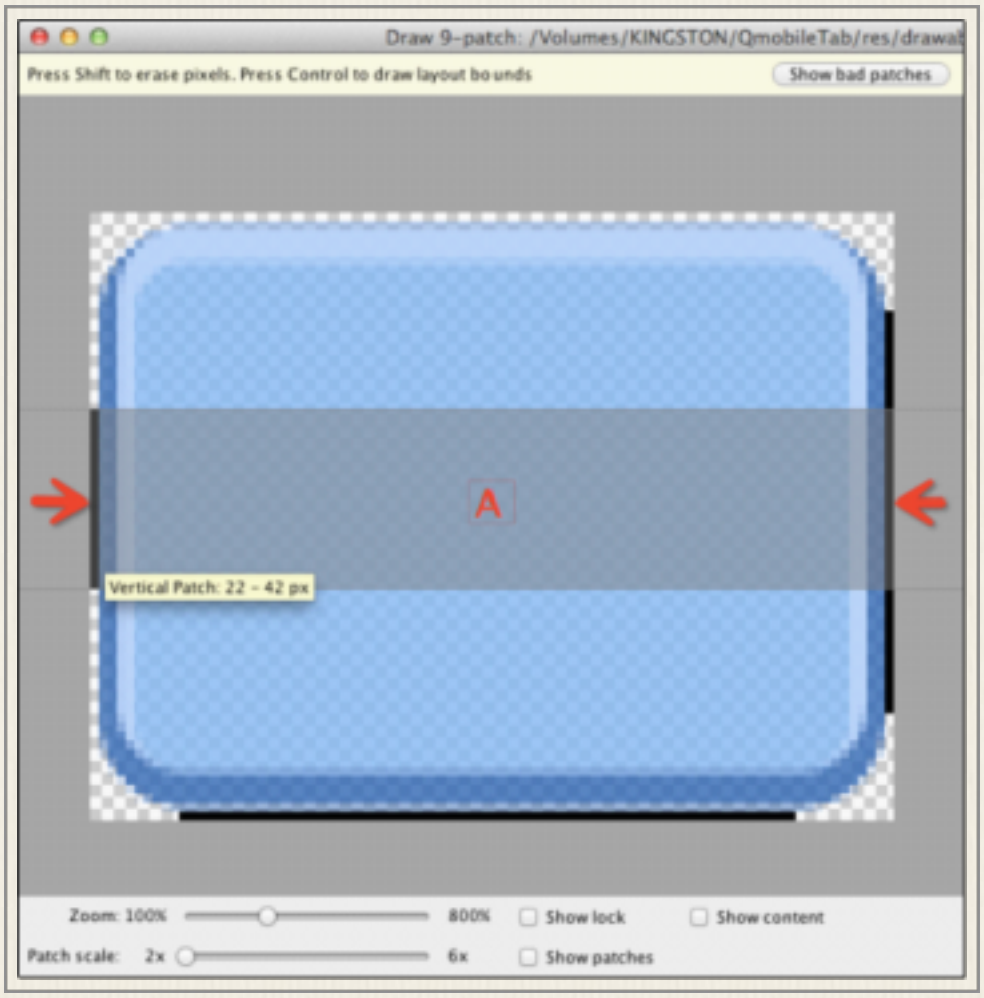

工具界面如下：



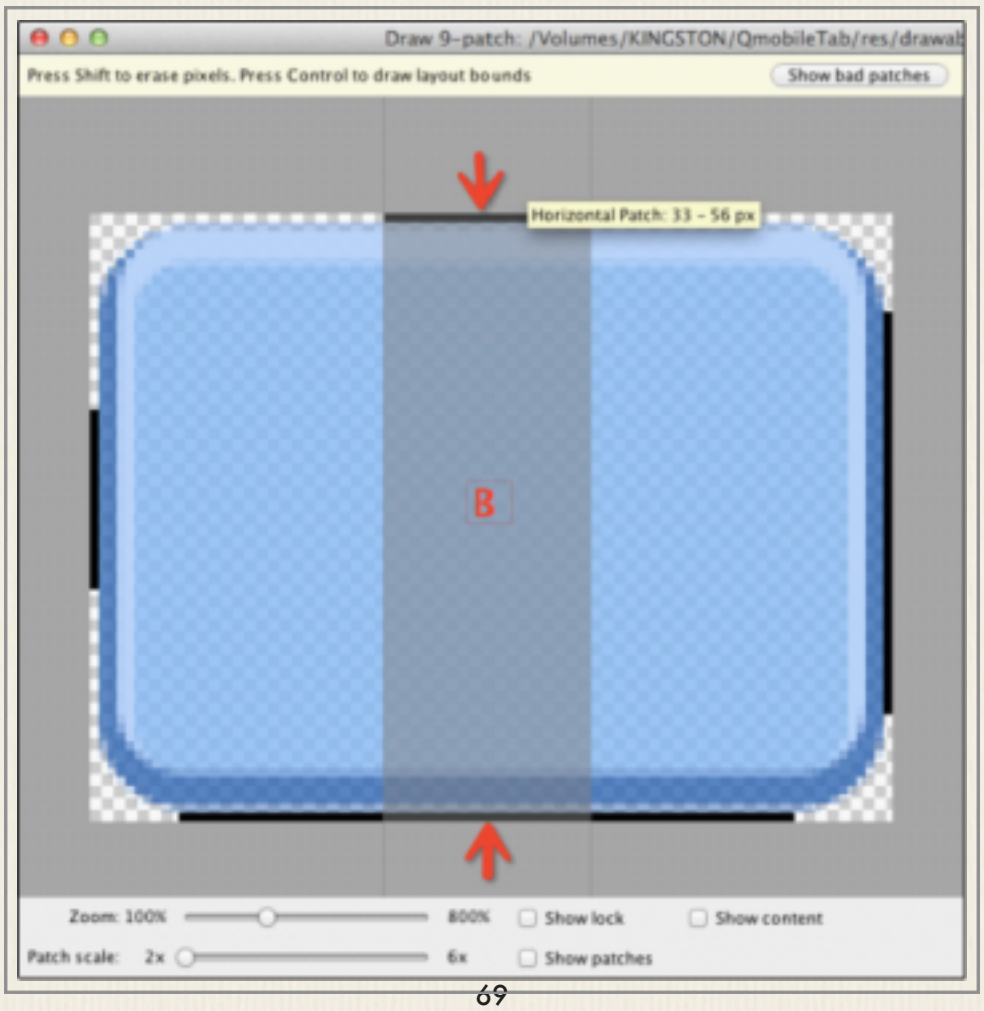
把图片拖拽到工具里面进行加点标记

在图片的左边（A区域左侧）和上边（B区域上方）进行标记，表示该区域可伸缩。

A区域:

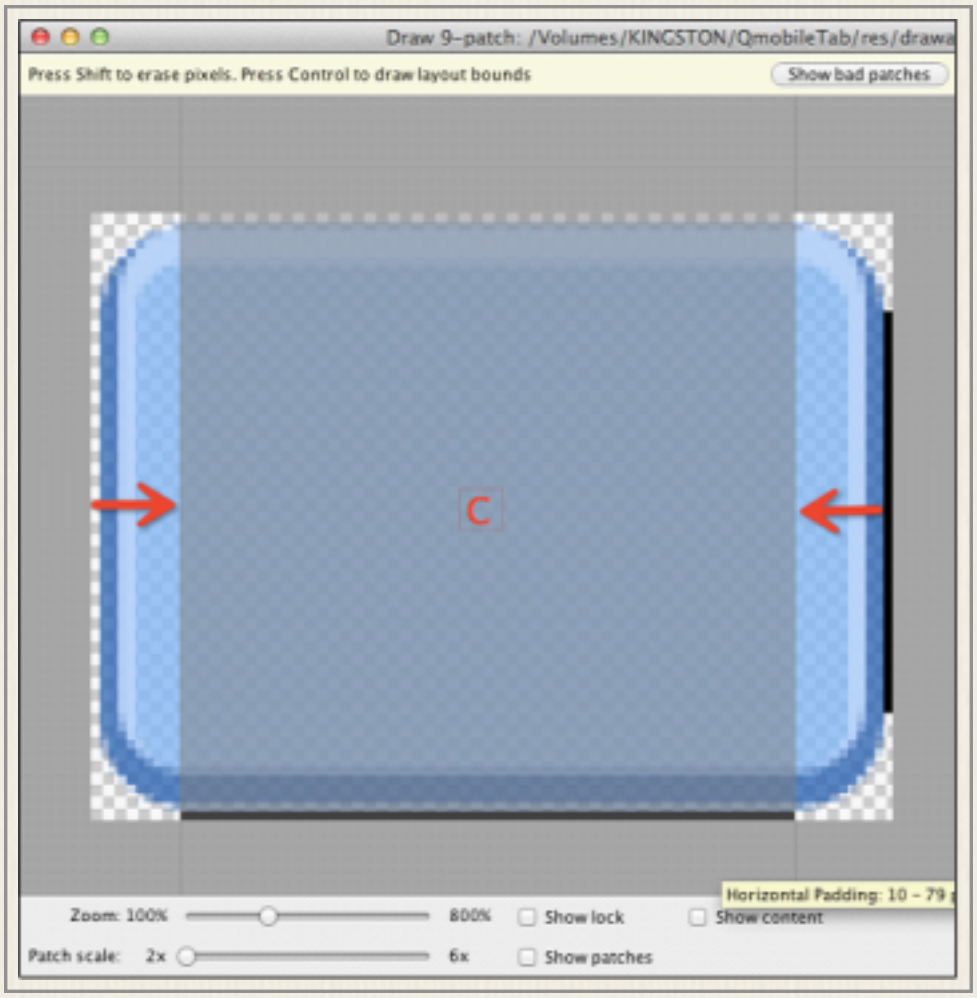



B区域:

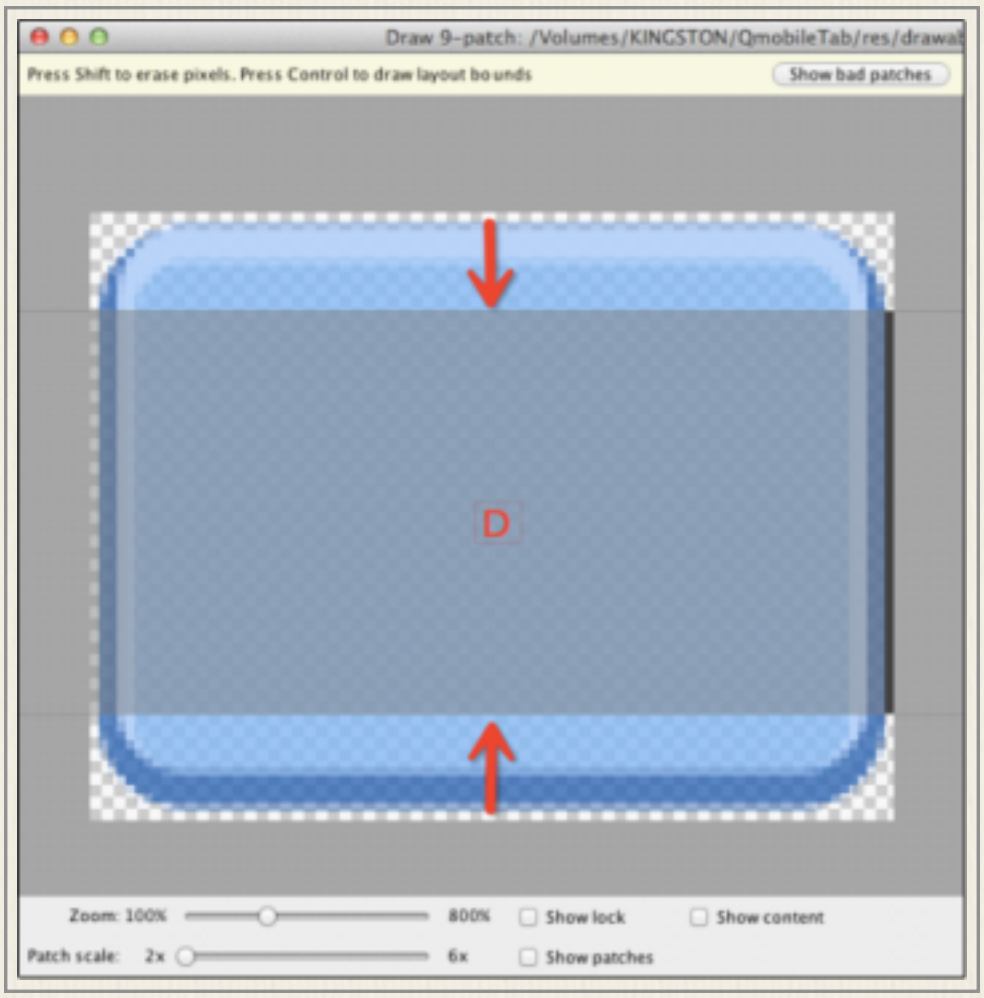


在图片的底边（c区域下方）和右边（D区域右侧）进行标记，表示该区域是内容。

C区域:

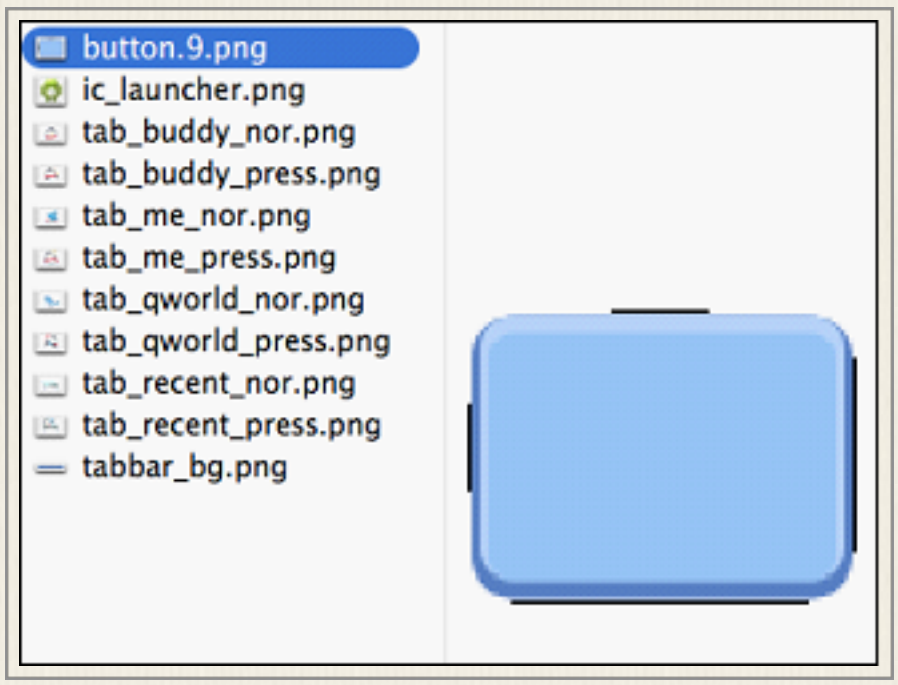



D区域:



标记完之后，文件在保存的时候会将改为.9.png格式。

使用时，在xml文件中设置android:background=”@drawable/button”。




关于点九的文章前面有同学介绍过：http://isux.tencent.com/android-ui-9-png.html

当然，如果你想偷懒的话，这里有个工具可以帮你点:)

http://romannurik.github.io/AndroidAssetStudio/nine-patches.html

## 四，总结

在做屏幕适配时，需要注意以下几个点：

1. 在定义xml布局的时候，使用dp单位，wrap_content，fill_parent(match_parent)
2. 为不同的屏幕密度提供不同的图片
3. 对于有某些规则(渐变)的图片使用点九图
4. 为特殊的屏幕类型(横竖屏)提供不同的布局

对于本节来说，是非常基础的知识，同时，这也是打造精品APP的必备知识，各位加油吧:P

原文链接:

腾讯ISUX (http://isux.tencent.com/learn-android-from-zero-session6.html)